دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پروژه ی مهندسی نرم افزار
سامانه رزرو نوبت آنلاین

دانشجو:
مهدیس صفری
9531051

استاد درس :
دکتر عبداله زاده بارفروش

خرداد 1398

1. ساختار سازمانی پروژه

نمودار ساختار سازمانی یک نمودار است که ساختار سلسله مراتبی یک سازمان را نشان می دهد.

ساختار سازمانی این پروژه به این صورت است که سیستم تحت نظارت صاحب سیستم و مدیر کنترل کیفیت، مدیریت می شود.

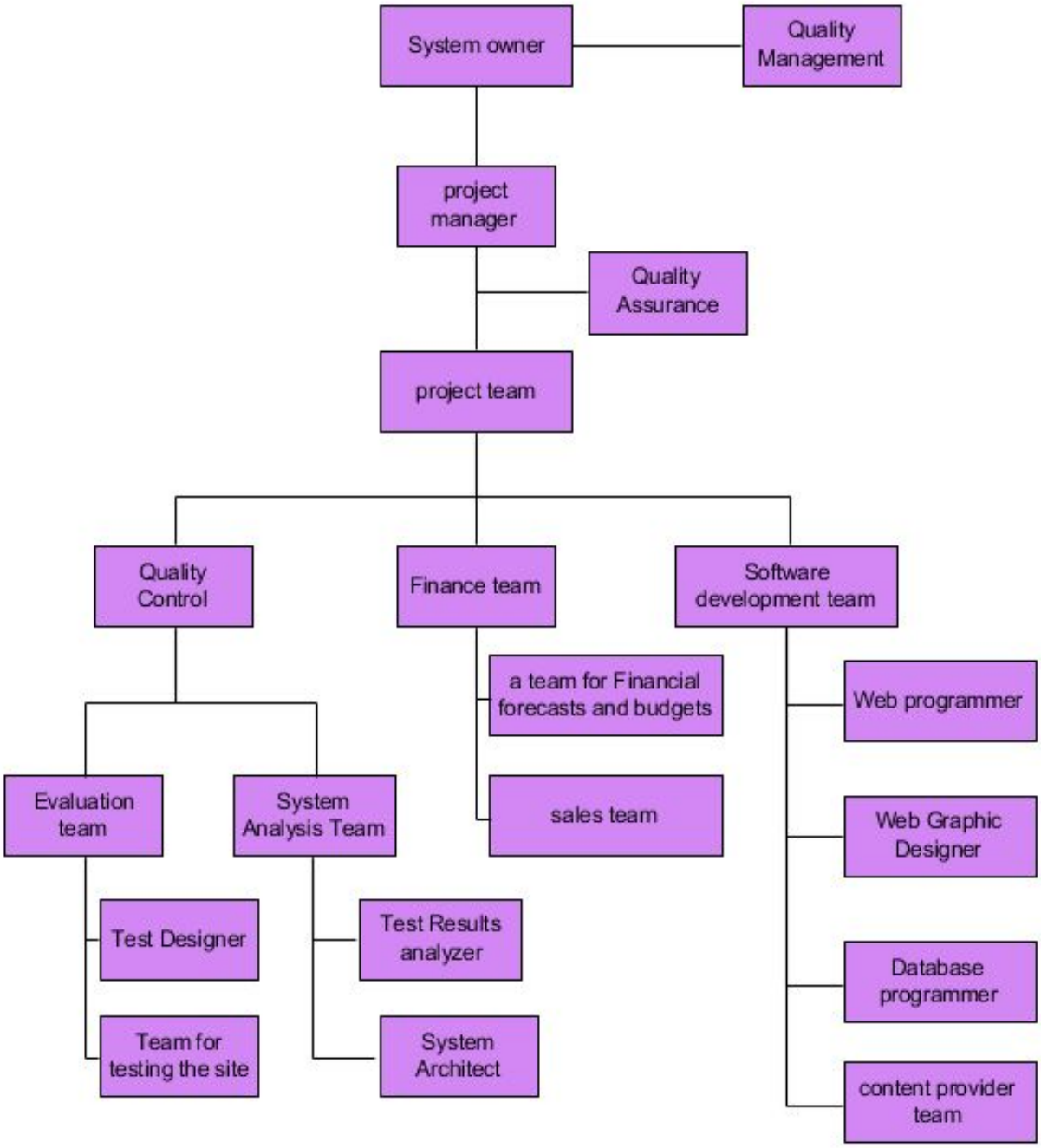
در این پروژه تیم توسعه نرم افزار،تیم امور مالی و تیم کنترل کیفیت سه تیم اصلی می باشد.

تیم توسعه ی نرم افزار شامل تیم تولید محتوا، برنامه نویس وب، برنامه نویس پایگاه داده و طراح گرافیکی سایت است. تیم تولید محتوا از سمت مدیر بیمارستان برنامه ی پزشکان بیمارستان را جمع آوری کرده و در اختیار مدیر سایت قرار میدهند.

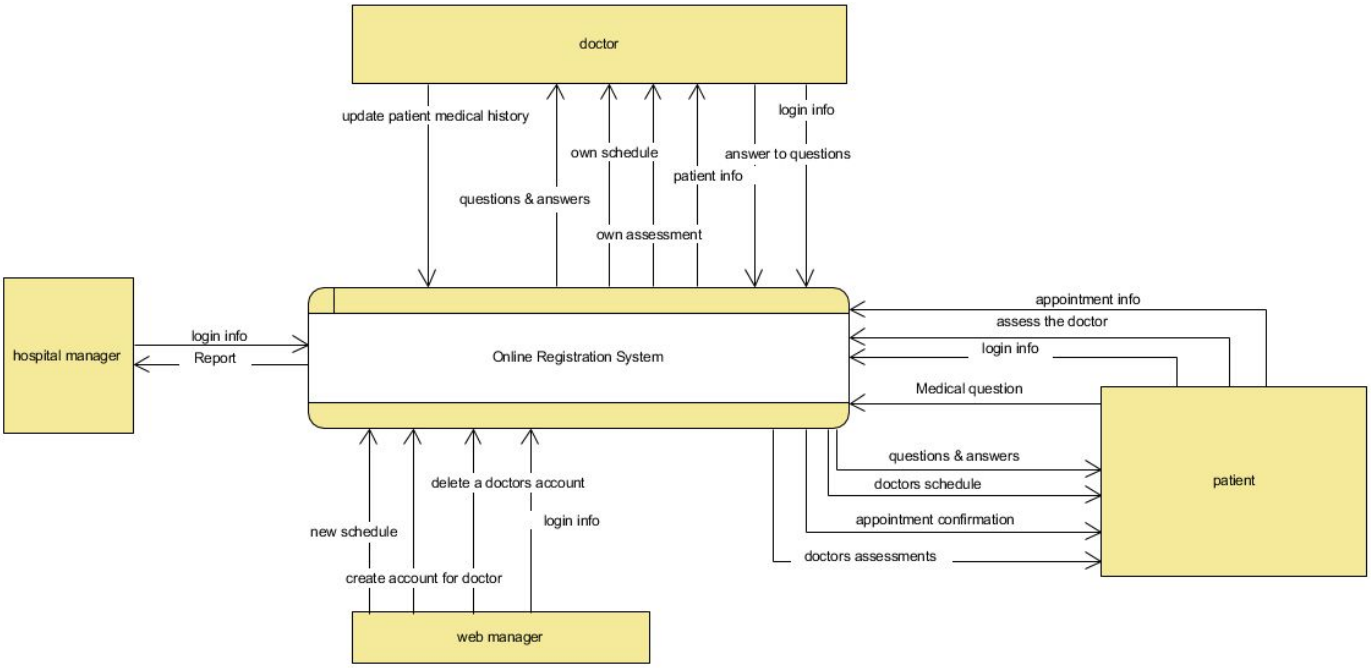
تیم امور مالی، امور مربوط به پیش بینی هزینه و مدیریت بودجه و همچنین فروش پروژه به بیمارستان ها را بر عهده دارد.

تیم کنترل کیفیت شامل دو زیر گروه آنالیز سیستم و ارزیابی سیستم می باشد.

تیم ارزیابی سیستم شامل طراحی و اجرای تست برای تضمین کیفیت سیستم و تیم آنالیز سیستم شامل آنالیز نتایج تست های انجام شده و معماری سیستم است.



Context diagram .2



3.لایه های مهندسی نرم افزار

در ادامه به توضیح مختصری از هر لایه می پردازیم و درباره ی هرکدام از این لایه ها به سوالات W5H2 پاسخ میدهیم.

1. Quality

بستری که کیفیت فعالیت های مختلف مهندسی نرم افزار را مشخص می کند و از آن پشتیبانی می کند.
شامل انتخاب استراتژی های تضمین کیفیت، استانداردهای و شاخص ها و معیارهای اندازه گیری و ... می شود.
روشهای مهندسی بر اساس نیازمندی های کیفیت سازمانی مشتری (راندمان، قابلیت اطمینان و غیره)، الزامات کیفیت توسعه (نگهداری، قابلیت مجدد و غیره)، کاربران (قابلیت استفاده، کارایی، و غیره) و غیره مورد نیاز است.
یک محصول باید مشخصات(specifications) خود را برآورده کند.

what	استراتژی های تضمین کیفیت، استانداردهای و شاخص ها و معیارهای اندازه گیری و ...
who	تیم ها Quality Control تیم ها Quality Assurance افرادی از تیم پروژه که وظیفه ی ارزیابی سیستم را دارند.
when	در حین تولید سیستم
why	برای تضمین کیفیت محصول و پشتیبانی از موفقیت پروژه
where	شرکت سازمانی که مدیریت پروژه را برعهده دارد و سیستم در آن توسعه داده می شود.
how	استفاده از مدل های کیفیتی مختلف یا طراحی تست برای ارزیابی تطابق سیستم با نیازمندی های سیستم صحیح سیستم بر اساس این تست ها
How much	تا زمانی که اصل good enough برآورده شود.

2. Processes

این لایه پایه اصلی مهندسی نرم افزار است. مدل فرایندی در این لایه مشخص می شود که به وسیله آن نقاط عطف و ترتیب انجام فعالیت های چارچوبی مشخص می شود که شامل فعالیت ها و وظایف مختلف است.به طور خلاصه، تمام فعالیت ها، اقدامات و وظایف مورد نیاز برای توسعه نرم افزار را پوشش می دهد.
مهمترین تصمیم در این مرحله انتخاب مدل فرایندی است.

what	مشخص کردن نقاط عطف و ترتیب انجام فعالیت های چارچوبی و انتخاب مدل فرایندی

who	مدیر پروژه و تیم توسعه نرم افزار
when	در حین تولید سیستم
why	برای کنترل سرمایه و مدیریت تیم توسعه نرم افزار در راستای رسیدن به کیفیت بهتر و همچنین رعایت time-b پروژه
where	شرکت سازمانی که مدیریت پروژه را برعهده دارد و سیستم در آن توسعه داده می شود.
how	با داشتن process framework
How much	تا پایان پروژه (رسیدن به کیفیت مورد نظر)

3. Methods

این لایه یک مجموعه گسترده ای از وظایف را شامل تجزیه و تحلیل نیازمندی ها، طراحی، برنامه نویسی، آزمایش و مرحله نگهداری را پوشش می دهد و شامل موارد زیر می شود.

شامل مجموعه ای از وظایف است که از طریق ارتباطات، تجزیه و تحلیل نیازها، تجزیه و تحلیل و مدل سازی طراحی، ساخت برنامه، آزمایش و پشتیبانی، آغاز می شود.

- انتخاب روش تحلیل نیازمندی ها(مبتنی بر سناریو، رفتاری و ..)
- انتخاب روش استخراج نیازمندی ها(مصاحبه، مشاهده)
- انتخاب روش طراحی(طراحی داده گرا، شی گرا و ...)
- انتخاب روش آزمایش نرم افزار(آزمایش سفید، سیاه و ..)
- و انتخاب های دیگر در مراحل مختلف نرم افزار

what	مجموعه گسترده ای از وظایف را شامل تجزیه و تحلیل نیازمندی ها، طراحی، برنامه نویسی، آزمایش رحله نگهداری
who	تیم تجزیه و تحلیل نیازمندی ها، تیم طراحی سیستم، تیم برنامه نویسی، تیم آزمایش سیستم
when	در حین تولید سیستم
why	برای داشتن برنامه مشخص و استاندارد که باعث می شود سیستم از نظر تکنیکی در مسیر درست کت کند.
where	شرکت سازمانی که مدیریت پروژه را برعهده دارد و سیستم در آن توسعه داده می شود.
how	داشتن رویکرد و متدولوژی مشخص
How much	تا پایان پروژه (رسیدن به کیفیت مورد نظر)

4. Tools

شامل ابزاری می شود که با استفاده از آنها می توان سیستم را به شکل خودکار یا نیمه خودکار از دو لایه ی قبل پشتیبانی کرد.

گاهی اوقات ابزارها به گونه ای متصل می شوند که دیگر ابزارها می توانند از اطلاعات ایجاد شده توسط یک ابزار استفاده کنند.این multi-usage معمولا به عنوان(Computer-Aided Software Engineering (CASE شناخته می شود. CAS شامل نرم افزار، سخت افزار و پایگاه داده مهندسی نرم افزار برای ایجاد نرم افزار مشابه Computer-Aided Design (CAD) برای سخت افزار است.

CASE در توسعه نرم افزارها از جمله تجزیه و تحلیل، طراحی، تولید کد، و اشکال زدایی و آزمایش کمک می کند.

نمونه هایی از ابزار CASE شامل ابزار نمودار، ابزار مستند سازی، ابزارهای مدل سازی فرآیند، ابزارهای تجزیه و تحلیل و طراحی، ابزارهای نرم افزاری سیستم، ابزارهای مدیریت پروژه، ابزارهای طراحی، ابزارهای نمونه سازی، ابزارهای مدیریت پیکربندی، ابزارهای برنامه نویسی، ابزار توسعه وب، ابزارهای تست، ابزارهای تعمیر و نگهداری ، ابزار تضمین کیفیت، ابزار مدیریت پایگاه داده و ابزارهای re-engineering.

what	ابزارهای برای کمک به جریان تجزیه و تحلیل، طراحی، تولید کد، و اشکال زدایی و آزمایش مانند ابزار نمودار مانند visual paradigm یا ابزار مدیریت پایگاه داده
who	اعضای تیم پروژه بسته به زمینه ای که در آن فعالیت دارند ممکن است از ابزار متفاوتی استفاده کنند.
when	در حین تولید سیستم
why	استفاده از یک زبان جامع در مستندسازی بخش های مختلف که موجب درک درستی از مستندات می د.
where	شرکت سازمانی که مدیریت پروژه را برعهده دارد و سیستم در آن توسعه داده می شود.
how	با یادگیری و استفاده از این ابزار در زمینه ی مربوطه
How much	از نظر زمانی تا پایان پروژه (رسیدن به کیفیت مورد نظر) و از نظر مالی به هزینه ابزار مورد نیاز گی دارد

4. رویکرد و متدولوژی پروژه

در این پروژه از رویکرد Agile و متدولوژی ASD استفاده میکنیم.
در ادامه با مقایسه ی رویکردها و متدولوژی رویکرد انتخابی دلیل این انتخاب را بیان می کنیم.
ابتدا به مقایسه ی سه رویکرد object oriented ،agile و structured می پردازیم.

	agile	object oriented	structured
معرفی کلی	روشِی برای ایجاد یک سیستم با طراحي، برنامه ریزی و برنامه نویسی تکرار شونده	رویکرد پایین به بالا ارتباط بین اشیاء مختلف در هنگام اجرای یک طراحی شی گرا وجود دارد.	رویکرد بالا به پایین معمولا در تجزیه و تحلیل ساختاری استفاده می شود
تمرکز	customer	object	process
استفاده ی مجدد	به قدر کافی	زیاد	کم
مناسب برای	پروژه های کوچک یا متوسط با پیچیدگی زیاد و تحویل سریع و افزایشی	پروژه های ریسکی بزرگ با امکان تغییر نیازمندی های کاربر	پروژه هایی که به صورت کامل تعریف شده اند و نیازمندی های کاربر تغییر نمی کند.
ریسک	کم	کم	زیاد
طراحی موازی با برنامه ویسی	دارد	دارد	ندارد
مشارکت کاربر	زیاد	در فازهای کمی مشارکت دارد	ندارد

این سیستم یک پروژه کوچک یا متوسط است که میخواهیم با کنترل ریسک بالا با استفاده از مشارکت کاربر محصولی مطابق با نیازمندی های کاربر داشته باشیم. با استفاده از رویکرد agile در این پروژه وظایف را به گام های کوچک با کمترین میزان برنامه ریزی تقسیم می کنیم به طوری که با برنامه ریزی های طولانی مدت درگیر نباشیم.
محصول به صورت مرحله ای در بسته های زمانی متوسط یک ماه به دست مشتری می رسند.

متدولوژی های Agile :

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method(DSDM)
- Scrum
- Crystal
- Feature Driven Development (FDD)

	ASD	FDD	DSDM	Srum	XP
رویکرد	Iterative	Iterative	Iterative	Iterative rements	Iterative rements
مدت زمان هر iterati	چهار تا هشت هفته	دو روز تا دو هفته	80% محصول در 20% زمان مورد نیاز	دو تا چهار هفته	یک تا شش هفته
اندازه ی تیم پروژه	خیلی کوچک (3 تا 9 عضو)	بیش از یک تیم با نای زیاد	تیم های مستقل با هر زه	هر اندازه	کوچک (کمتر از 20 عضو)
اندازه پروژه مناسب	کوچک	پروژه های پیچیده	هر اندازه	هر اندازه	کوچک
مشارکت کاربر	مشارکت از طریق ار مکرر محصول	مشارکت از طریق رش ها	مشارکت از طریق انتشار ر محصول	مشارکت از طریق ک محصول	مشارکت زیاد با کاربر
مستندسازی	فقط مستندسازی پایه ای	اهمیت دارد	دارد	فقط مستندسازی پایه ای	فقط مستندسازی پایه ای
شیوه های عمده	Learning cycles	UML diagrams	Prototyping, Feasibility and siness study	Sprint, Product and Sprint Backlogs, Scrum meetings	User Stories, Test, Driven Development, Refactoring, Pair programming

ویژگی های ASD:

- ❑ Mission focused

نیازمندی ها گاهی به طور کامل مشخص نیستند اما mission کلی سیستم مشخص است و مرزهای سیستم را تعیین می کند و تصمیمات را جهت دهی می کند.

- ❑ Feature based
- ❑ Iterative
- ❑ Time-boxed planning

زمان مورد نیاز برای انجام یک activity/task مشخص شده است.

- ❑ Risk driven
- ❑ Change tolerant
- ❑ Component based

در این پروژه قصد داریم با یک تیم برنامه نویسی محدود، با انتشار مکرر سیستم در دوره های زمانی کوتاه ،نتیجه را بررسی کرده و تغییرات لازم را اعمال کنیم .

حال به معرفی این متدولوژی می پردازیم.

: ASD

این متدولوژی شامل سه فاز می باشد که در ادامه، اهداف هر فاز بیان شده است. در طی این سه فاز تمامی فعالیت های تحلیل مقدماتی، تحلیل تفصیلی، طراحی، پیاده سازی، تست و قرارگیری سیستم در محیط کاربر انجام شده و فقط فاز مراقبت و نگهداری را شامل نمی شود، بنابراین تمام فعالیت های چرخه ی عمومی ایجاد نرم افزار را پوشش نمی دهد.

: initiation project

در این فاز کارهای مربوط به تحلیل مقدماتی انجام می شود. اهداف پروژه تعیین شده و تخمین ها برای اندازه و زمینه ی پروژه زده می شود. محدودیت ها و ریسک های موجود شناسایی شده و تیم های ایجاد تشکیل می شوند. نیازمندی های سطح بالا استخراج شده و نهایتا معیارهای موفقیت برای سیستم مشخص می گردند.

سه فاز بعدی این متدولوژی به صورت تکرار شونده اجرا می شوند.

: planning cycle adaptive

در این فاز زمان اتمام پروژه و هر یک از سیکل های ایجاد تخمین زده می شوند. مولفه هایی که باید پیاده سازی شوند تعیین شده و به سیکل ها اختصاص می یابند. زمانبندی تکرارها نیز انجام می شود. بنابراین هدف کل این فاز برنامه ریزی می باشد و این برنامه در اول هر تکرار دقیق می شود.

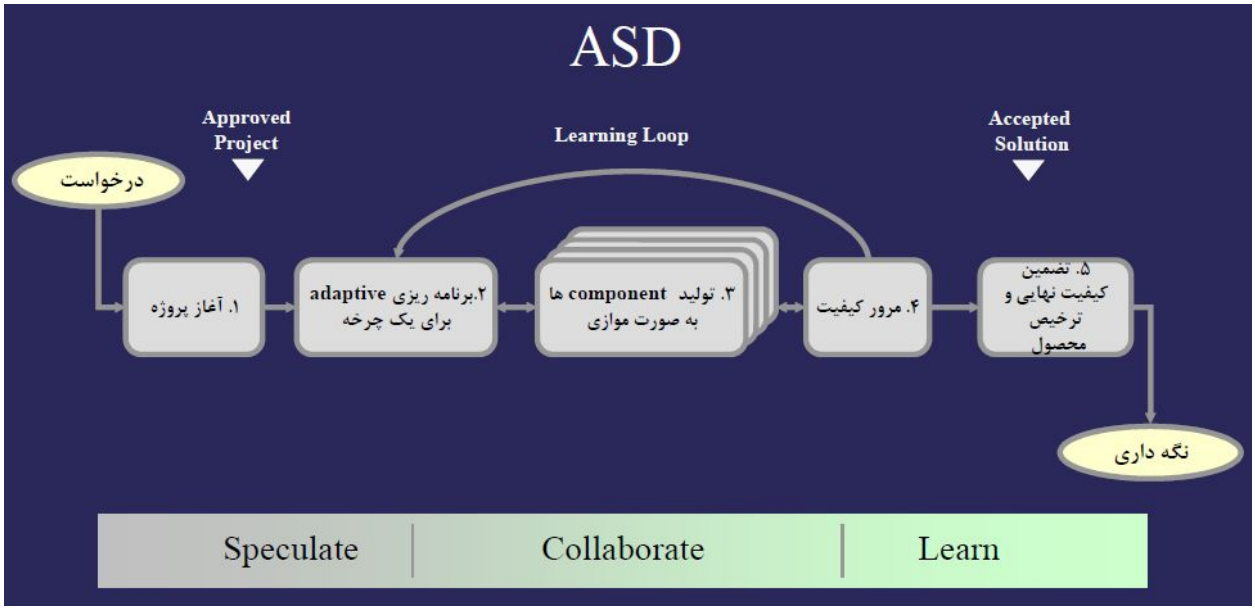
: component concurrent engineering

در این مرحله طراحی و پیاده سازی مولفه های تخصیص داده شده به هر یک از سیکل ها به صورت همروند انجام می شود.

: review quality

در این فاز مرور مولفه های تولید شده به صورت گروهی انجام می گیرد و مشکلاتی که با آنها مواجه شده اند برطرف می گردند.

آخرین فاز release and A/Q final می باشد. در این مرحل سیستم نهایی تولید شده ارزیابی شده و پس از آن در محیط کاری کاربر قرار می گیرد.



6. process framework

ابتدا task های هر کدام از فاز های ASD را مشخص می کنیم.

1. initiation project
- تحلیل مقدماتی شامل مشخص کردن اهداف پروژه
- تخمین اندازه ی پروژه
- شناسایی محدودیت ها و ریسک های موجود شده
- تشکیل تیم هایی برای ایجاد سیستم
- شناسایی نیازمندی های سطح بالا و معیارهای موفقیت سیستم
- تعیین scope و not scope های سیستم

2. planning cycle adaptive
- مشخص کردن time-box پروژه و تعداد iteration های مورد نیاز
- مشخص کردن time-box هر iteration پروژه
- مشخص کردن component هایی که باید تولید شوند
- تخصیص component ها به هر iteration

3. component concurrent engineering

- طراحی مولفه های تخصیص داده شده به هر یک از سیکل ها به صورت همروند
- پیاده سازی مولفه های تخصیص داده شده به هر یک از سیکل ها به صورت همروند

4. review quality
- مرور مولفه های تولید شده به صورت گروهی
- انجام تست
- ارزیابی نتیجه
- برطرف کردن مشکل

5. release and A/Q final
- ارزیابی سیستم
- release کردن سیستم

از فعالیت های همیشگی چارچوب فرایند میتوان به نکات زیر اشاره کرد :
- مدیریت پیکربندی سیستم
- مدیریت پروژه نرم افزار
- بررسی فنی رسمی
- تضمین کیفیت سیستم
- اندازه گیری سیستم
- مدیریت ریسک

7. ورودی و خروجی process framework

- initiation project

ورودی	خروجی
اهداف پروژه	اندازه پروژه شناخت محدودیت ها و ریسک ها تشکیل تیم پروژه نیازمندی ها و معیارهای موفقیت سیستم scope و not scope

- planning cycle adaptive

ورودی	خروجی

نیازمندی های سیستم محدودیت ها و ریسک ها	time-box پروژه component ها component های هر iteration تعداد iteration ها و time-box هرکدام

- component concurrent engineering

ورودی	خروجی
component های هر iteration تعداد iteration ها و time-box هرکدام	طراحی مولفه های هر iteration پیاده سازی مولفه های هر iteration

- review quality

ورودی	خروجی
مولفه های هر iteration تست	نتیجه ی تست ارزیابی نتیجه تست برطرف شدن مشکل

- release and A/Q final

ورودی	خروجی
سیستم تست	تضمین کیفیت نهایی ترخیص محصول

8. ذی نفعان سیستم

ذی نفع یعنی افراد و گروه‌هایی که به نوعی در موفقیت یا شکست یک کسب و کار، سهیم هستند.

- ❖ کاربران سیستم :
  - ➢ بیماران
  - ➢ پزشکان
  - ➢ مدیر بیمارستان
- ❖ تیم تولید محتوا
- ❖ شرکت نرم افزاری :
  - ➢ مدیر پروژه
  - ➢ مدیر شرکت
  - ➢ مدیر سایت
  - ➢ تیم ارزیابی
  - ➢ تیم توسعه نرم افزار

برنامه نویس وب

برنامه نویس پایگاه داده

طراح گرافیکی وب سایت

- ➢ تیم تحلیل سیستم
- ➢ تیم امور مالی

9. نیازمندی های سیستم به همراه مدل نیازمندی ها

Actorهای سیستم:

- پزشک
- بیمار
- مدیر بیمارستان
- مدیر سایت

پزشک :

1. ورود به حساب کاربری
2. خروج از حساب کاربری
3. مشاهده برنامه یک ماه آتی
4. مشاهده ی ارزیابی خود
5. مشاهده ی پرونده ی پزشکی مراجعه کننده
6. پاسخ به سوالات در قسمت پرسش و پاسخ

7. مشاهده ی پرسش و پاسخ های موجود
8. به روز کردن پرونده ی پزشکی مراجعه کننده

بیمار:

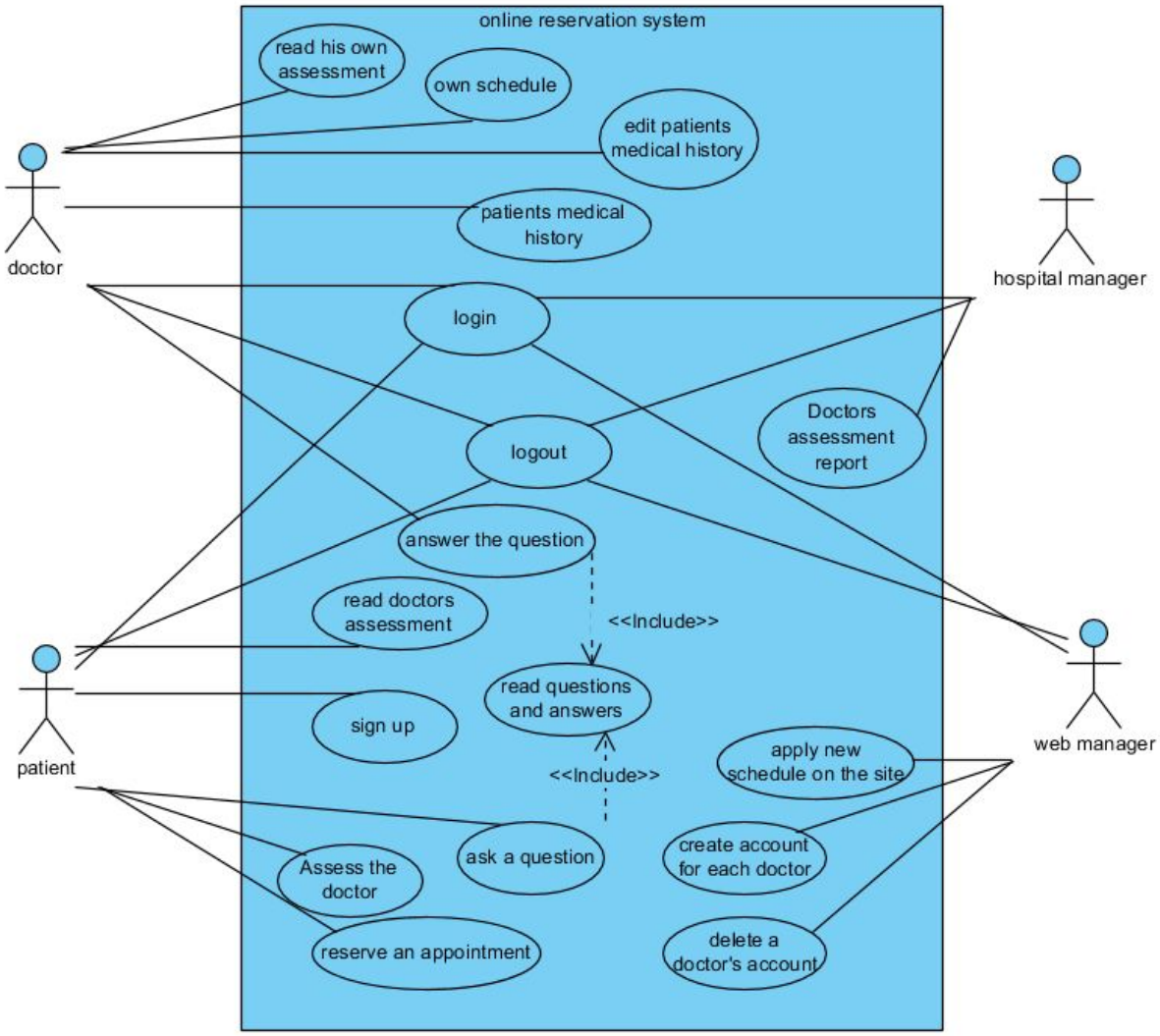
1. ورود به حساب کاربری
2. خروج از حساب کاربری
3. ایجاد حساب کاربری
4. مشاهده ی ارزیابی پزشک
5. پرسش سوال در قسمت پرسش و پاسخ
6. مشاهده ی پرسش و پاسخ های موجود
7. ارزیابی پزشک
8. انتخاب بخش و پزشک مورد نظر
9. رزرو نوبت

مدیر بیمارستان:

- ورود به حساب کاربری
- خروج از حساب کاربری
- دریافت گزارش ارزیابی پزشکان

مدیر سایت:

1. ورود به حساب کاربری
2. خروج از حساب کاربری
3. به روز رسانی برنامه پزشکان
4. ایجاد حساب کاربری برای پزشکان
5. حذف حساب کاربری پزشک



10.دسته بندی نیازمندی ها

در این قسمت نیازمندی های سیستم بر اساس +FRUPS دسته بندی می شوند.

- functionality
  - همه ی نیازمندی هایی که در قسمت قبل برای actor ها تعریف شدند.
- Usability
  - وجود راهنمای کار با سیستم

- بازخورد گرفتن از کاربر
- فهرست الفبایی
- به روز بودن اطلاعات سایت
- Reliability
- حداقل کردن خطا در بازیابی اطلاعات کاربر در صورت بروز خطا
- Performance
- سرعت بارگذاری بالا
- مدیریت همزمانی درخواست چندین کاربر
- مثلا ممکن است چندین بیمار همزمان درخواست رزرو یک نوبت را داشته باشند.
- Support
- با مرورگر موبایل سازگار باشد
- در صفحه موبایل صفحات وب بهم نریزند.
- تست و تصحیح آسان سیستم

11. مستند SRS

1. معرفی

1.1. هدف

این وبسایت برای نوبت دهی آنلاین یک بیمارستان استفاده می شود. از قابلیت های دیگر سایت این است که کاربران در نقش بیمار میتوانند از قسمت مشاهده ی ارزیابی پزشک ها در تصمیم گیری خود برای انتخاب پزشک استفاده کنند، ارزیابی خود از پزشک را ثبت کنند و در قسمت پرسش و پاسخ، پاسخ سوالات پزشکی خود را بیابند.

پزشک ها با استفاده از این سایت می توانند برنامه ی یک ماه آتی خود را مشاهده کنند، به پرونده ی پزشکی بیمار مراجعه کننده و ارزیابی بیماران از خود دسترسی داشته باشند و پاسخ سوالات بدهند.

مدیر بیمارستان نیز می تواند از این ارزیابی ها در مدیریت بهتر بیمارستان استفاده کند.

1.2. مخاطبین این مستند

- مدیر پروژه
- سرمایه گذار پروژه
- مشتری سیستم مانند مدیر بیمارستان
- توسعه دهندگان سیستم

1.3. محدوده ی پروژه

محدوده ی کاربران سیستم پزشکان و کاربران عادی(بیماران)، مدیر بیمارستان، مدیر سایت می باشد.

امکانات این سیستم :

- رزرو نوبت آنلاین
- ایجاد پرونده برای بیماران
- پرسش و پاسخ با پزشکان
- ارزیابی بیماران از پزشک

آنچه برای این سیستم در نظر گرفته نشده است:

- پرداخت آنلاین حق ویزیت
- دسترسی بیماران به پرونده ی پزشکی خود
- کنسل کردن نوبت رزرو شده چه از طرف بیمار و چه از طرف پزشک
- پاسخ سوالات بخش پرسش و پاسخ توسط بیمار
- طرح پرسش در بخش پرسش و پاسخ توسط پزشک
- رزرو نوبت توسط کاربر غیر بیمار
- ارزیابی توسط پزشک

2. شرح کلی

2.1. چشم انداز محصول

کاربران شامل بیمار، پزشک، مدیر بیمارستان و مدیر سایت از طریق وب سایت از خدمات سیستم استفاده می کنند.

همه ی کاربران برای استفاده از خدمات سایت باید در سیستم عضو باشند. عضویت کاربران معمولی توسط خودشان و سایر کاربران توسط مدیر سایت انجام می شود.

قابلیت هایی که برای استفاده ی کاربر معمولی(بیمار) در نظر گرفته شده :

- رزرو نوبت پس از مشاهده ی برنامه ی یک ماه آتی پزشک مورد نظر
- طرح پرسش در قسمت پرسش و پاسخ با پزشکان
- ارزیابی پزشک و استفاده از نظر دیگران

قابلیت هایی که برای استفاده ی کاربر پزشک در نظر گرفته شده :

- مشاهده برنامه ی یک ماه آتی خود
- مشاهده ارزیابی بیماران از خود
- پاسخ به سوالات در قسمت پرسش و پاسخ با پزشکان
- مشاهده ی پرونده ی پزشکی بیمار مراجعه کننده و به روز رسانی آن

قابلیت هایی که برای استفاده ی کاربر مدیر سایت در نظر گرفته شده :

- به روز رسانی برنامه پزشکان
- ایجاد حساب کاربری برای پزشکان و مدیر بیمارستان
- حذف حساب کاربری پزشک

مدیر بیمارستان نیز می تواند گزارش ارزیابی پزشکان را دریافت کند و در مدیریت بهتر بیمارستان خود لحاظ کند.

این سایت برای مصارف عمومی بوده و محصول های مشابه آن وجود دارد و به صورت مستقل و نه جایگزین ارائه می شود.

2.2. قابلیت های محصول

- رزرو نوبت آنلاین
- دسترسی پزشک به پرونده ی پزشکی بیمار مراجعه کننده
- پرسش و پاسخ با پزشکان
- ارزیابی بیماران از پزشک

2.3. کاربران سیستم

یک دسته از عمده ترین کاربران این سیستم بیماران ب هستند که میخواهند در بیمارستان مورد نظر، از یک پزشک در بخش مورد نظرشان وقت بگیرند. این کاربران همچنین می توانند سوالات پزشکی خود را در بخش مرتبط پرسیده و یا نظر خود را راجع به پزشکان بیان کنند.

دسته ی دیگر از کاربران این سیستم پزشکان هستند. این کاربران میتوانند سوالات پزشکی پرسیده شده در صفحه ی بخش بیمارستان که در آن فعالیت دارند پاسخ دهند. به علاوه می توانند ارزیابی و برنامه ی یک ماه آتی مربوط به خود را مشاهده کنند.

کاربر دیگر این سیستم مدیر بیمارستان است. این کاربر می تواند ارزیابی پزشکان را مشاهده کند.

مدیر وب سایت اطلاعات سیستم و پایگاه داده را در اختیار دارد و می تواند آن را اصلاح کند به علاوه می تواند یک پزشک را حذف کند.همچنین برنامه ی دقیق بیمارستان را به روز رسانی می کند.

همه ی افراد به نوعی مي توانند کاربر این سیستم باشند.

2.4. محیط عملیاتی

کاربران می توانند از طریق مرورگر های خود (web browser) به سایت دسترسی داشته باشند. بنابراین هدف طراحی یک نرم افزار cross-platform است که بتواند در سیستم عامل های

مختلف از قبیل Mac OS ، Linux ، Microsoft windows ، قابلیت اجرا داشته باشد. در واقع ما بیشتر بر روی مرورگر ها تمرکز خواهیم کرد .

از آن جایی که وب سایت وابسته به سیستم عامل خاصی نیست ، بنابراین به پلتفرم سخت افزاری جدای از سخت افزار هایی که این سیستم عامل ها بر روی آن ها سوار شده اند نیاز ندارند.

در مورد محدودیت های سخت افزاری در اجرای کار میتوان گفت، بیشتر به میزبان اشتراکی یا سروری که خریداری خواهیم کرد، وابسته ایم ؛ مانند فضای دیسک و پهنای باندی که سرور در اختیار ما قرار میدهد.

2.5. محدودیت های طراحی و پیاده سازی

اجرای موازی یک درخواست به وسیله چند کاربر مختلف می تواند باعث ایجاد ناهماهنگی در ثبت اطلاعات شود. به عنوان مثال؛ ممکن است دو کاربر به طور همزمان درخواست ثبت یک زمان ملاقات واحد را داشته باشند و تیم باید بتواند سیاست های لازم را برای ثبت درست اطلاعات در پایگاه داده داشته باشد.

تعداد تراکنش های بالای کاربران با وب سایت و ارائه درخواست های مختلف از سوی کاربران ممکن است باعث ایجاد محدودیت در زیرساخت های وب سایت شده و باعث از دسترس خارج شدن سیستم از شبکه شود. تیم توسعه باید بتواند از زیرساختی استفاده کند که احتمال از دسترس خارج شدن سیستم در آن کم باشد.

ایجاد امنیت بالا برای حفظ اطلاعات کاربران , از دیگر محدودیت های تیم توسعه بوده و تیم باید از سازوکاری برای ایجاد نرم افزار استفاده نماید که امنیت کاربران حفظ شده و اطلاعات آنها فاش نشود. با توجه به اینکه سیستم در دست توسعه به صورت وب سایت میباشد , نمایش صحیح آن در مرورگرهای مختلف از محدودیت های تیم توسعه به شمار می رود و تیم باید قابلیت نمایش وب سایت در مرورگرهای مختلف همچنین مرورگرهای موبایل را لحاظ کند.

3. نیازمندی های عملیاتی

پزشک :

- ورود به حساب کاربری
- خروج از حساب کاربری
- مشاهده برنامه یک ماه آتی
- مشاهده ی ارزیابی خود
- مشاهده ی پرونده ی پزشکی مراجعه کننده
- پاسخ به سوالات در قسمت پرسش و پاسخ
- مشاهده ی پرسش و پاسخ های موجود
- به روز کردن پرونده ی پزشکی مراجعه کننده

بیمار:

- ورود به حساب کاربری
- خروج از حساب کاربری
- ایجاد حساب کاربری
- مشاهده ی ارزیابی پزشک
- پرسش سوال در قسمت پرسش و پاسخ
- مشاهده ی پرسش و پاسخ های موجود
- ارزیابی پزشک
- انتخاب بخش و پزشک مورد نظر
- رزرو نوبت

مدیر بیمارستان:

- ورود به حساب کاربری
- خروج از حساب کاربری
- دریافت گزارش ارزیابی پزشکان

مدیر سایت:

- ورود به حساب کاربری
- خروج از حساب کاربری
- به روز رسانی برنامه پزشکان
- ایجاد حساب کاربری برای پزشکان
- حذف حساب کاربری پزشک

4. نیازمندیهای واسط خارجی

4.1. واسط های کاربری

رابط کاربر نمای وبسایت را که کاربر با آن تعامل دارد، تعیین می کند. برای بالا بردن جذابیت وبسایت، فاکتورهای ساده بودن برای استفاده، پاسخ کوتاه در زمان کوتاه، استفاده از المان های واضح جهت درک بهتر و سازگاری بالا در تمام صفحات نمایش رابط کاربری در نظر گرفته می شود.

این وبسایت از رابط کاربری گرافیکی بهره می برد. این رابط کاربری، رابط کاربری ساده ای است و امکان تعامل با سیستم را فراهم می کند.

عناصر بصری مانند ترکیب بندی،رنگ بندی، تصویر سازی، فونت و... بعنوان اجزای رابط کاربری باید علاوه بر زیبایی و خلاقیت، کارایی و خاصیت تعاملی داشته باشند.

در طراحی رابط کاربری وب سایت از Flat Design با الگوی طراحی z استفاده می شود. طراحی تخت را یک روش ساده گرا می نامند که بیشتر روی کارآمدی تاکید دارد تا این که به واقعیت نزدیک باشد. از مزایای این طراحی سادگی و شفافیت، کاهش حجم صفحات و افزایش سرعت بارگذاری، درک آسان مطالب سایت برای کاربران، آسانی طراحی سایت واکنش گرا و همچنین افزایش سرعت پروژه است. الگوی طراحی z برای طراحی های ساده و با عناصر کلیدی اصلی کم مناسب میباشد.

صفحات اصلی وب سایت که بین کاربران مشترک است:

- صفحه ی log in
- صفحه ی راهنمای وب سایت

از دید بیمار:

- صفحه ی sign up
- صفحه ی انتخاب بخش و پزشک مورد نظر (صفحه ی رزرو نوبت)
- صفحه ی مشاهده ی برنامه پزشک و ارزیابی او (امکان انتخاب نوبت خالی و ارزیابی پزشک را دارد)
- صفحه ی پرسش و پاسخ (تنها امکان پرسش سوال را دارد)

از دید پزشک:

- صفحه ی مشاهده ی برنامه و ارزیابی خود (تنها امکان مشاهده ارزیابی خود را دارد)
- صفحه ی پرسش و پاسخ (تنها امکان پاسخ به سوال را دارد)
- صفحه ی مشاهده ی پرونده ی پزشکی بیمار (با امکان به روز رسانی پرونده ی بیمار)

از دید مدیر بیمارستان:

- صفحه ی گزارش ارزیابی پزشکان

از دید مدیر سایت:

- صفحه ی مدیریت برنامه پزشکان (با امکان حذف برنامه و حساب کاربری پزشک)
- صفحه ی ایجاد حساب کاربری برای پزشکان

4.2. واسط های سخت افزاری

سایت طراحی شده در این پروژه روی انواع رایانه های شخصی، موبایل، IPad و تبلت ها سازگار است و از تمامی این دستگاه ها میتوان به این وبسایت به خوبی دسترسی پیدا کرد.

برای انتقال داده های ورودی از موس و کیبورد در pc ها و از صفحات لمسی یا صفحه کلید ها در موبایل ها یا تبلت ها استفاده می شود و مانیتور یا صفحه گوشی و تبلت اطلاعات را نمایش میدهد.

4.3. واسط های نرم افزاری

این سیستم برای ذخیره و بازیابی اطلاعات مورد نیاز مانند اطلاعات حساب کاربران و برنامه پزشکان و ... نیازمند تعامل با پایگاه داده میباشد.

5. نیازمندی های غیر عملیاتی

Usability

وجود راهنمای کار با سیستم

بازخورد گرفتن از کاربر

فهرست الفبایی

به روز بودن اطلاعات سایت

Reliability

حداقل کردن خطا در بازیابی اطلاعات کاربر در صورت بروز خطا

Performance

سرعت بارگذاری بالا

مدیریت همزمانی درخواست چندین کاربر. مثلا ممکن است چندین بیمار همزمان درخواست رزرو یک نوبت را داشته باشند.

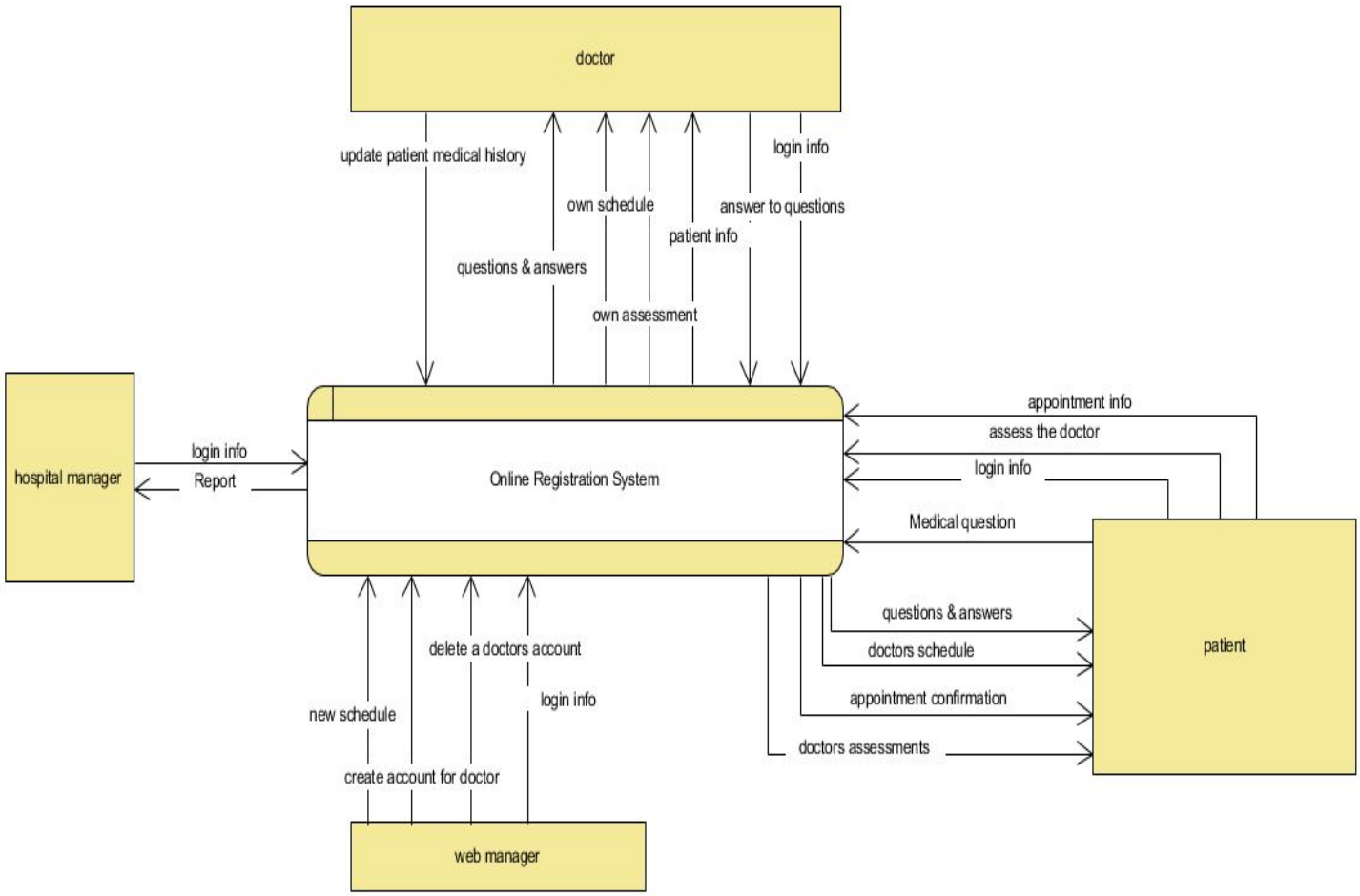
Support

با مرورگر موبایل سازگار باشد (در صفحه موبایل صفحات وب بهم نریزند.)

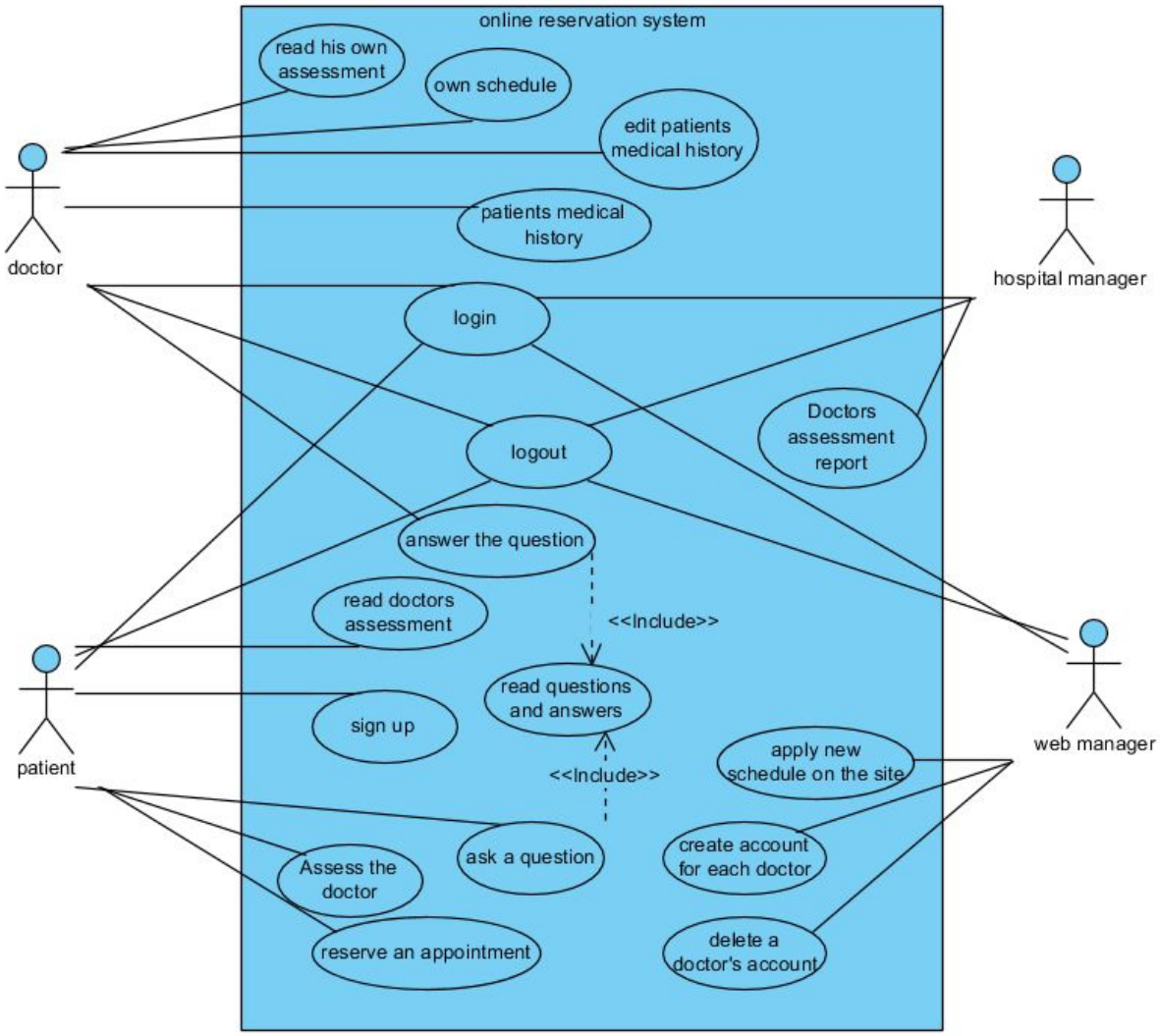
تست و تصحیح آسان سیستم

12.مدل های طراحی سیستم

- **Context Diagram**

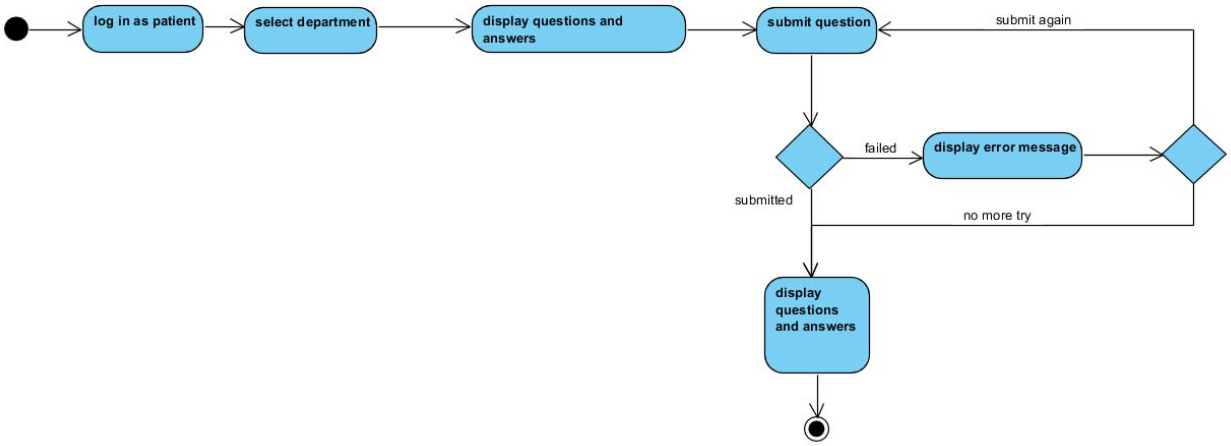


- **UseCase Diagram**

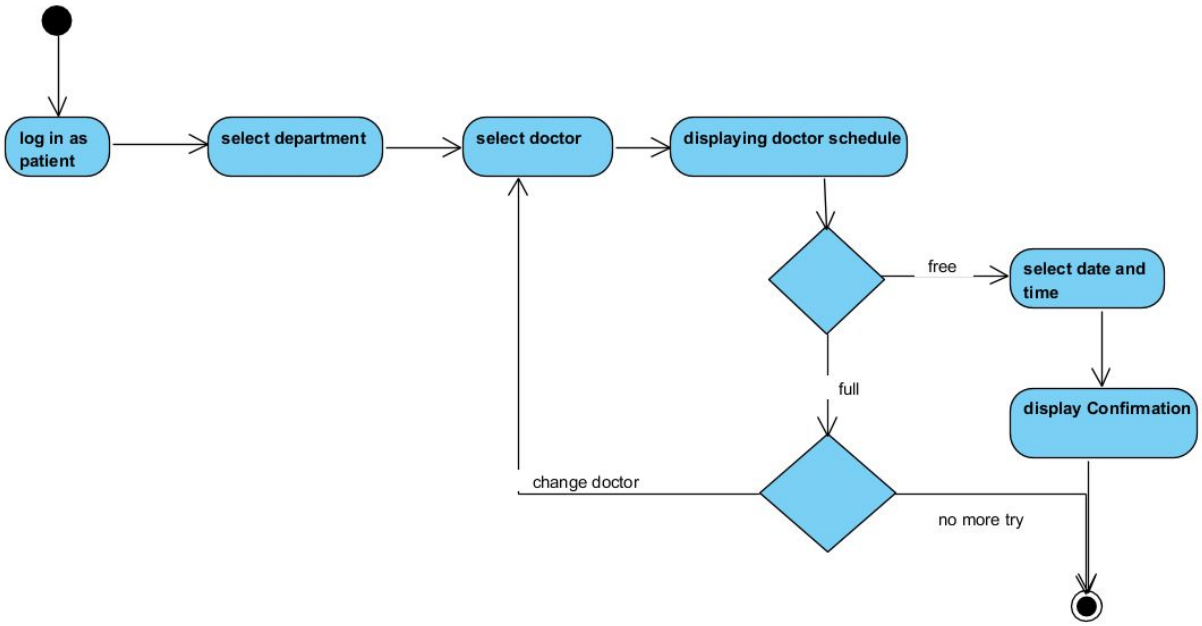


- **Activity Diagram**

نمودار فعالیت پرسش

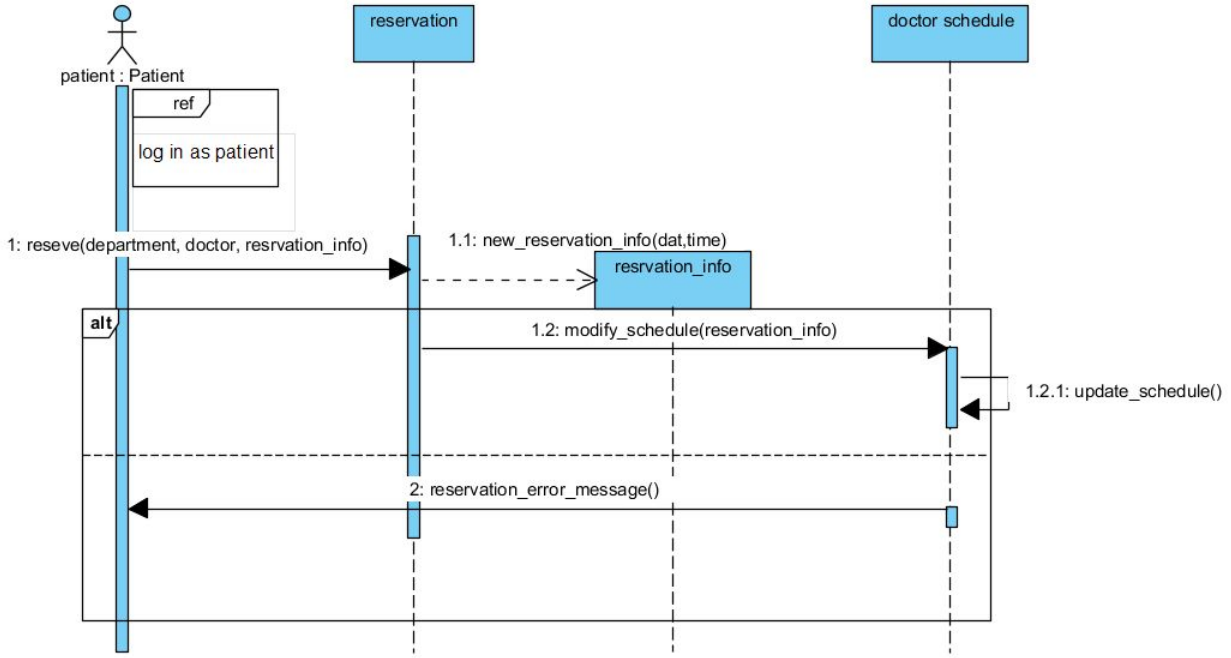


نمودار فعالیت رزرو نوبت

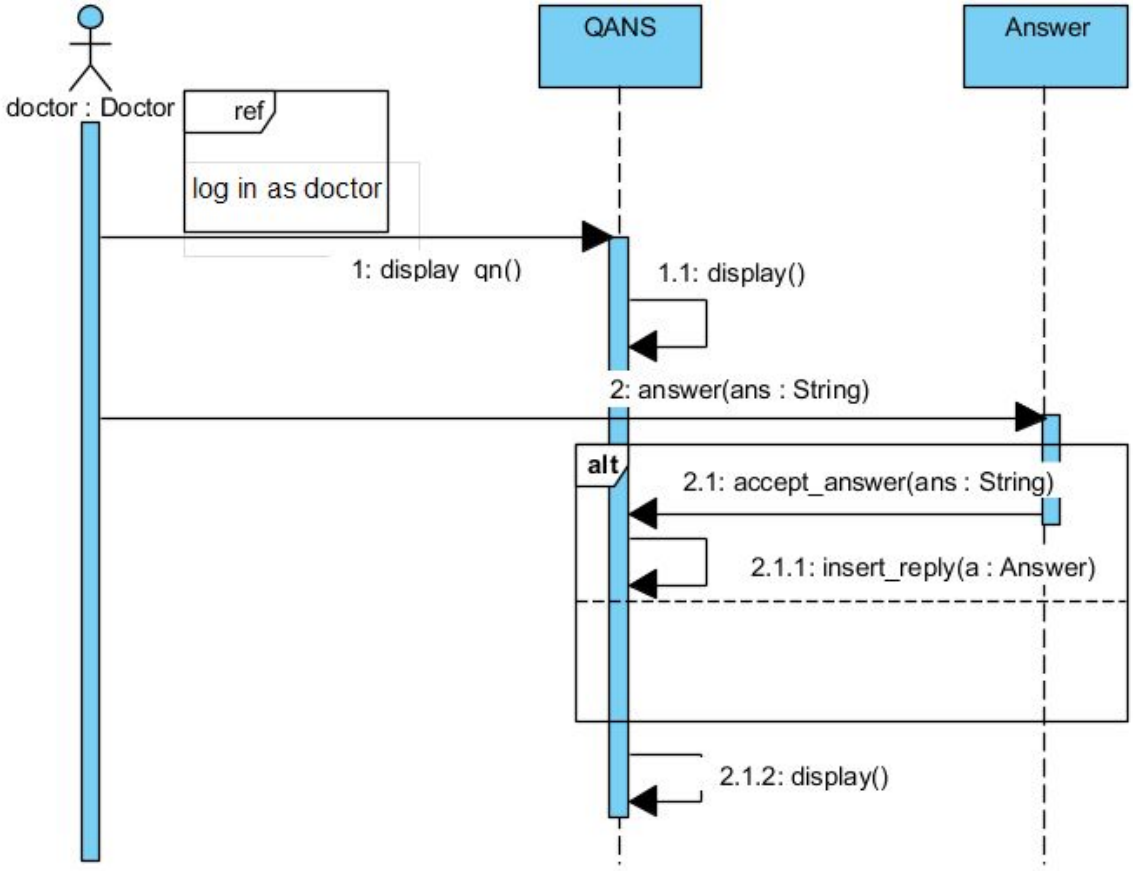


- **Sequence Diagram**

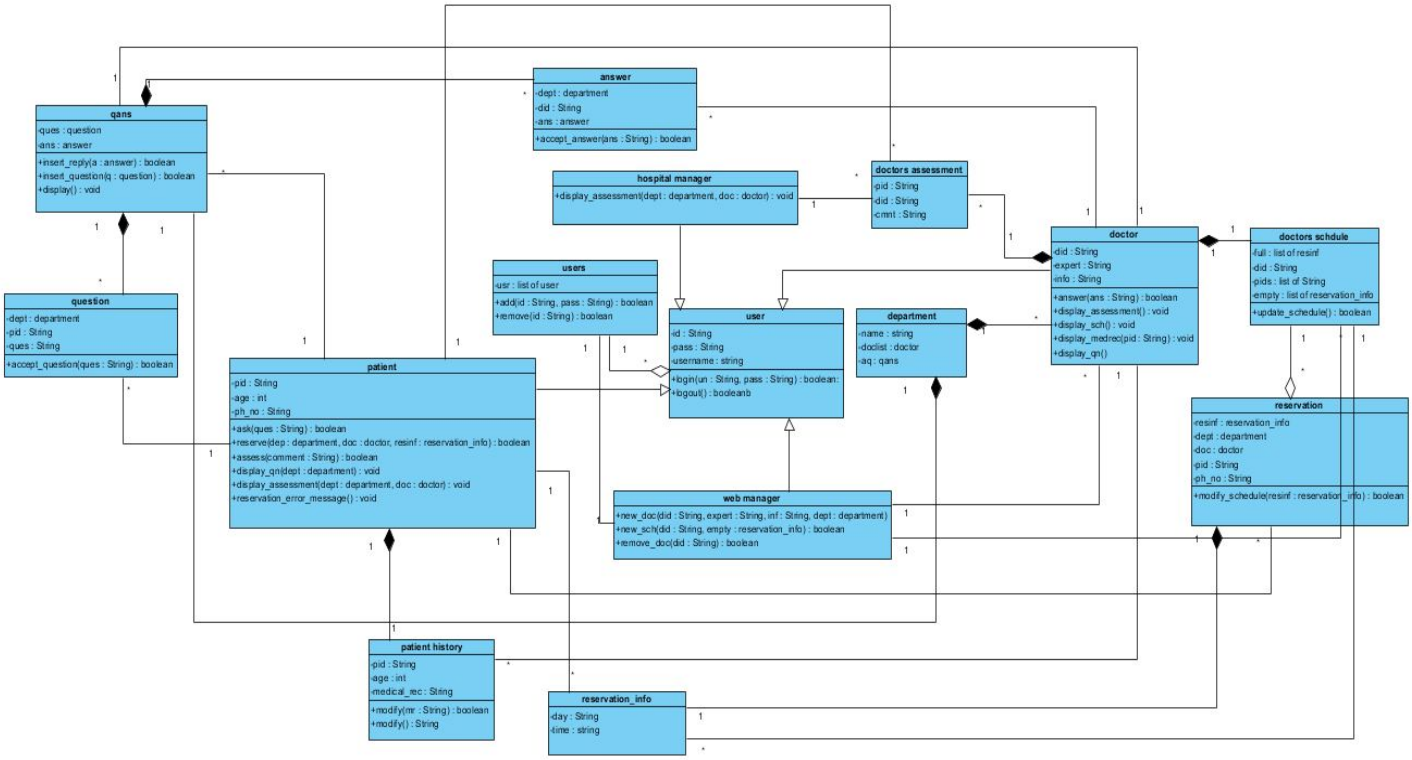
نمودار توالی رزرو نوبت



نمودار توالی پاسخ به سوالات



- **Class Diagram**



13. علت انتخاب مدل های استفاده شده،

برای مدلسازی پروژه از UML برای مدل سازی استفاده شد که طبق بررسی های انجام شده نمودار های توصیه شده برای متدولوژی agile ، طبق منبع ذکر شده در پایان همین بخش مدل های زیر بودند.

در بین نمودار هایی که UML برای مدلسازی ارائه می دهد نمودار های زیر برای مدل سازی استفاده شدند :

- context diagram

این نمودار سیستم را به عنوان یک high-level process ، ارتباط سیستم با سایر نهادهای خارجی را نشان داده و scope و boundary سیستم را در به سادگی نمایش می دهد.

از آن جایی که دانش فنی خاصی برای فهم این نمودار لازم نیست، می تواند به مخاطبان گسترده ای از جمله ذینفعان،data analysts،business analyst و توسعه دهندگان برای فهم بهتر سیستم کمک کند.

- Use case diagram

یک use case رفتار سیستم یا بخشی از سیستم را از دید موجودیت های خارجی سیستم نشان می دهد و با تعیین گام های موردنیاز برای دستیابی به یک هدف معین، مشخص می کند که کاربران به چه نحوی با

سیستم در تعامل هستند و به این صورت دید کلی ای از پروژه می دهد.

همچنین نیازمندی های عملکردی سیستم را نشان می دهد و در طی فرآیند جمع آوری نیازهای سیستم برای تشخیص اینکه سیستم چه قابلیت هایی را دارد که برای تطابق با نیازمندی های کاربر استفاده می شود

در این نمودار تعیین what و عدم توجه به how و نحوه ی پیاده سازی اهمیت دارد.

همچنین Use case در تولید test case نیز مورد استفاده قرار می گیرند.

- sequence diagram

از جمله نمودارهای رفتاری است که ترتیب تعاملات بین اجزا را در حین اجرای یک task برای مثال یک سناریوی use case مشخص می نماید.

برخلاف نمودار usecase که به what می پردازد بر روی how تمرکز دارد و نشان می دهد که برای انجام هر usecase کدام کلاس ها باید با هم ارتباط برقرار کنند و در این ارتباط function هایی استفاده می شوند. در نتیجه به پیاده سازی کد پروژه کمک زیادی می کند.

استفاده از این نمودار طراحی سیستم را نیز بهبود می دهد. برای مثال، تشخیص روابط پیچیده بین object ها را آسان تر می کند.

از این نمودار میتوان در integration testing استفاده کرد.

- activity diagram

این نمودار از نمودار های رفتاری سیستم است و بیشتر بر انتقال جریان کنترل و توالی عملیات ها بین object های مختلف تمرکز دارد. برای مدل کردن جنبه ی dynamic سیستم و بررسی جریان سیستم از نمودار استفاده شده است.

- class diagram

در این نمودار اجزای مورد نیاز ، صفت ها و عملیات های هر کلاس و نوع روابط بین آن ها نمایش داده می شود . این نمودار برای طراحی کلاس های لازم برای پیاده سازی سیستم در فاز component concurrent engineering استفاده می شود.

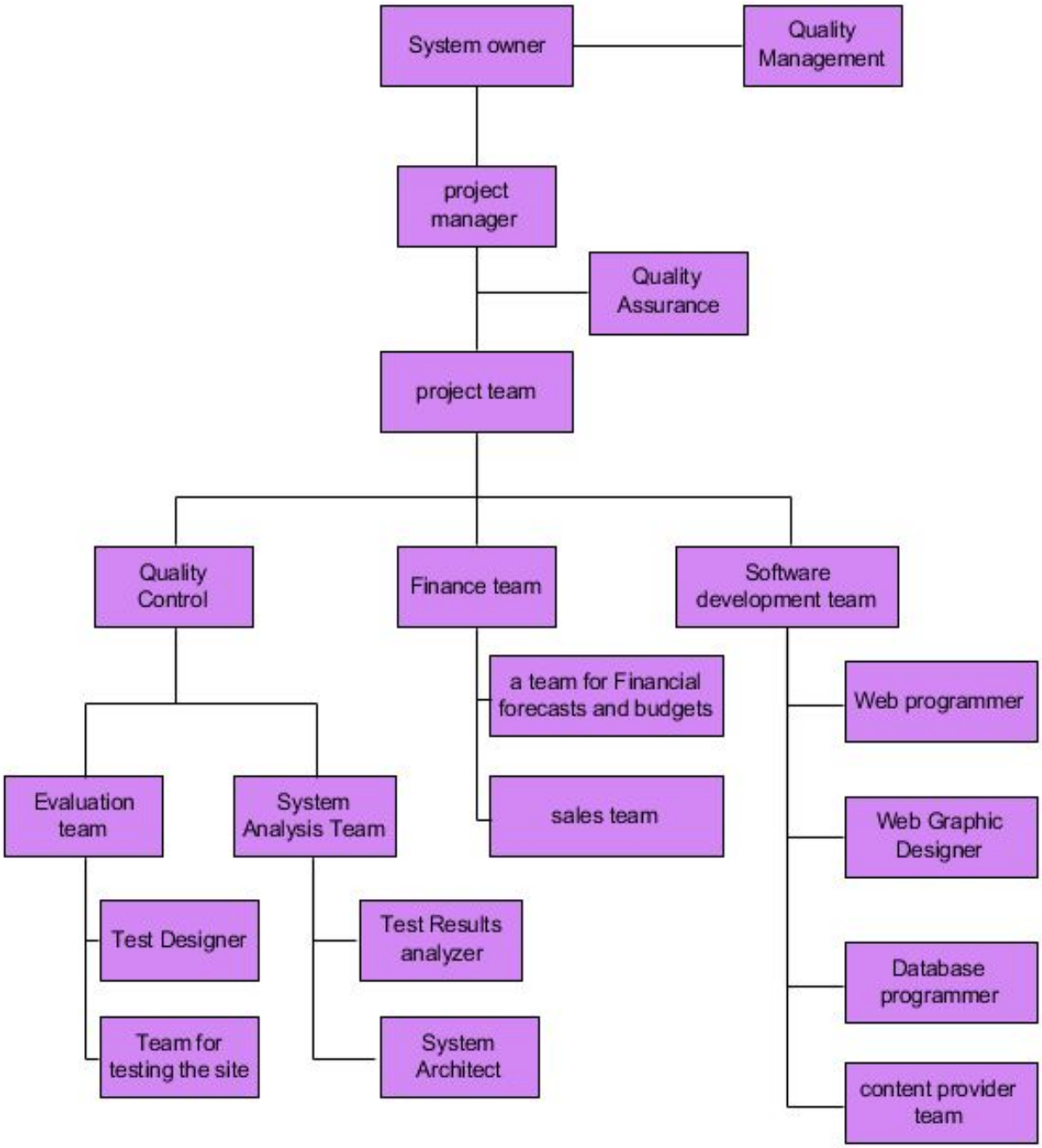
در این مستند سعی شد از هر مدل نمونه های کافی (برای نشان دادن فهم نمودار) ضمیمه شود. برای پیاده سازی این پروژه لازم است نمودار فعالیت و توالی برای همه ی فعالیت ها و سناریو ها رسم شوند.

منابع:

اسلایدهای uml و

http://agilemodeling.com/essays/umlDiagrams.htm

14.جایگاه و نقش Quality control و Quality assurance


System owner
Quality Management
project manager
Quality Assurance
project team
Quality Control
Finance team
Software development team
a team for Financial forecasts and budgets
Web programmer
Evaluation team
System Analysis Team
sales team
Web Graphic Designer
Test Designer
Test Results analyzer
Database programmer
Team for testing the site
System Architect
content provider team

QA, QC and Testing in software development process

طبق تعریف کنترل کیفیت در کتاب پرسمن، کنترل کیفیت شامل مجموعه ای از اقدامات مهندسی نرم افزار است که کمک می کند تا اطمینان حاصل شود که هر محصول مطابق با اهداف کیفیت آن است.در واقع مدل ها بررسی می شوند تا اطمینان حاصل شود که آنها کامل و سازگار هستند.

وظایف کنترل کیفیت

- Review
  - بازنگری نیازمندی ها
  - بازنگری طراحی
  - بازنگری کد
  - بازنگری طرح توسعه
  - بازنگری طرح تست
  - بررسی موارد تست
- Testing
  - تست واحد
  - تست مجتمع
  - تست سیستم
  - تست پذیرش

در صورتی که تضمین کیفیت اینگونه تعریف می شود که تضمین کیفیت متشکل از مجموعه ای از حسابرسی و گزارش عملکرد است که اثربخشی و کامل بودن اقدامات کنترل کیفیت را ارزیابی می کند.

هدف از تضمین کیفیت این است که کارمندان مدیریت و فنی را با اطلاعات لازم در مورد کیفیت محصول آگاه کند و همچنین به آنها بینش و اطمینانی بدهد که اقدامات آنها برای دستیابی به کیفیت محصول درست عمل می کند.

البته، اگر اطلاعات ارائه شده از طریق تضمین کیفیت مشکلات را تشخیص دهد، این مسئولیت مدیریت کیفیت است که مشکلات را مشخص کند و منابع لازم براي حل مسائل كيفي به كار گيرد.

وظایف تضمین کیفیت

- یک طرح SQA برای یک پروژه تهیه می کند.
  - این طرح به عنوان بخشی از برنامه ریزی پروژه توسعه یافته و توسط تمام سهامداران مورد بررسی قرار می گیرد. فعالیت های تضمین کیفیت انجام شده توسط تیم مهندسی نرم افزار و گروه SQA توسط این طرح اداره می شود. این طرح ارزیابی هایی را که باید انجام شود، ممیزی ها و بررسی های انجام شده، استانداردهای قابل اجرا برای پروژه، روش های گزارش گیری و ردیابی خطا، محصولات کاری که توسط گروه SQA تولید می شود، و بازخورد هایی که به تیم نرم افزاری ارائه می شود، را تعیین می کند.
- مشارکت در توسعه توصیف فرآیند نرم افزار پروژه.
  - تیم نرم افزاری یک فرایند را برای کاری که انجام می دهد انتخاب می کند. گروه SQA توصیف فرایند را برای انطباق با سیاست های سازمانی، استانداردهای نرم افزار داخلی، استانداردهای اعمال شده خارجی و سایر قسمت های طرح پروژه نرم افزار بررسی می کند.
- بررسی فعالیت های مهندسی نرم افزار برای بررسی انطباق با فرآیند نرم افزار تعریف شده.
  - گروه SQA شناسایی، اسناد و مدارک و انحراف از روند را دنبال می کند و تأیید می کند که اصلاحات انجام شده است.
- حسابرسی محصولات تعیین شده توسط نرم افزار برای بررسی انطباق با آنهایی که به عنوان بخشی از فرایند نرم افزار تعریف شده است.
  - گروه SQA محصولات مورد بررسی را بررسی می کند؛ شناسایی، اسناد و مدارک و انحرافات را دنبال می کند. تأیید می کند که اصلاحات انجام شده است و به طور دوره ای نتایج کار خود را به مدیر پروژه گزارش می دهد.
- اطمینان حاصل می کند که انحراف در برنامه های نرم افزاری و محصولات کار مستلزم و با توجه به یک روش مستند به کار گرفته می شود.
  - ممکن است در طرح پروژه، شرح فرآیند، استانداردهای قابل اجرا یا محصولات کار مهندسی نرم افزار با انحرافات مواجه شویم
- هرگونه عدم انطباق و گزارش به مدیریت ارشد را ثبت می کند.
  - آیتم های عدم انطباق ردیابی می شوند تا زمانی که حل شوند.

	Quality Assurance	Quality Control
Definition	QA is the implementation of processes, methodologies and standards that ensure that the software developed will be up to the required quality standards.	QC is the set of activities that are carried out to verify the developed product meets the required standards.
Target	QA focuses on the improvement of process and methodologies used to develop product.	QC focuses on the improvement of the product by identifying the bugs and issues.
Orientation	It is process oriented.	It is product oriented.
Nature of process	QA is preventive process as it establishes the methods which prevent the bugs.	QC is corrective process as it focuses on identifying the bugs and getting them fixed.
Verification vs Validation	Quality Assurance is a verification activity that verifies you are doing the right thing in the right manner.	Quality assurances is a validation activity that validates the product against the requirements.
Who	All the persons involved in the project starting from the requirement.	It is the responsibility of Quality Control inspector or the testing team that finds the issues.
Tools and Techniques	Defining Processes, Quality Audit, Selection of Tools, Training.	Defining Processes, Quality Audit, Selection of Tools, Training.
Examples	Examples of quality assurance activities include process checklists, process standards, process	Examples of quality control activities include inspection, deliverable peer

https://reqtest.com/testing-blog/quality-assurance-vs-quality-control-differences-2/

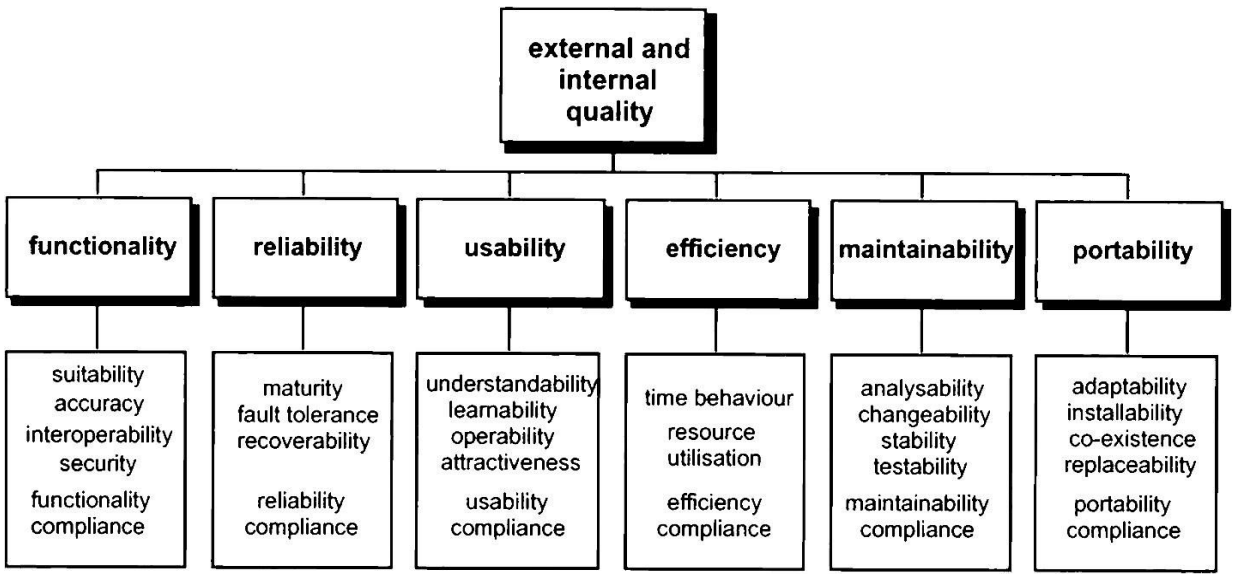
15. طراحی فریم مهندسی کیفیت

برای این کار باید از یکی از استانداردهای موجود استفاده کنیم. برای انتخاب استاندارد مناسب باید بررسی کرد که کدام مدل استاندارد نیازمندی های مورد نظر ما را بهتر پوشش می دهد. برای این کار از جدول زیر استفاده می کنم.

Factor/Model	*ISO*	*McCall*	*Boehm*	*FURPS*	*Dromey*
Efficiency	80%	70%	55%	25%	50%
Integrity	25%	100%	25%	25%	0%
Reliability	70%	65%	50%	65%	50%
Usability	63%	60%	60%	73%	0%
Correctness	0%	100%	0%	0%	50%
Maintainability	73%	68%	64%	0%	50%
Testability	25%	78%	53%	0%	0%
Changeability	25%	83%	42%	0%	0%
Interoperability	25%	100%	25%	0%	0%
Reusability	0%	100%	0%	0%	50%
Portability	78%	67%	61%	0%	50%
Functionality	86%	0%	0%	71%	50%
Understandability	0%	0%	25%	0%	0%
Human Engineering	0%	0%	75%	25%	0%
Total	**39.27%**	**63.67%**	**38.19%**	**20.34%**	**25.00%**

طبق مقایسه ی انجام شده مدل ISO با توجه به فاکتورهای مورد اهمیت در این پروژه مناسب تر است.

مدل ISO 9126 :



Maintainability، Efficiency، Usability، Reliability، Functionality و Portability ابعاد کیفیتی این مدل استاندارد را شامل می شوند.

16. metric و measurement های ابعاد کیفی مورد نظر

factor	measurement	metric	indicator
accuracy	میزان صحت عملکرد ها	نسبت تعداد عملکرد های صحیح به تعداد کل عملکرد ها	بیش از 95%
Compliance	میزان رعایت قوانین صنعتی یا دولتی	درصد موارد رعایت شده از روی چک لیست تهیه شده	بیش از 80%
security	میزان دسترسی غیرمجاز به سیستم	نسبت تعداد دسترسی های غیرمجاز ناموفق به تعداد کل دسترسی های غیرمجاز در یک بازه ی زمانی	نزدیک به 0
maturity	میزان failure سیستم	نسبت تعداد درخواست های Fail شده به تعداد کل درخواست ها در یک بازه زمانی	کمتر از 10%
fault tolerance	میزان مدیریت و بازیابی سیستم در برابر failure	نسبت تعداد failure های مدیریت شده به تعداد کل failure ها در یک بازه زمانی	بیش از 85%
recoverability	میزان بازیابی سیستم در زمان failure	نسبت تعداد failure های بازیابی شده به تعداد کل failure ها در یک بازه زمانی	بیش از 70%
understandability	میزان سهولت فهم عملکردهای سیستم	نسبت تعداد کاربرانی که از راهنما استفاده می کنند به تعداد کل کاربرها در یک بازه زمانی	کمتر از 20%
learnability	میزان تلاش یادگیری برای	مدت زمانی که طول می کشد تا کاربر از یکی از قابلیت	کمتر از 15min

	های سیستم استفاده کند	کاربران مختلف	
بیش از $60	درصد رضایت کاربر از زیبایی سایت با توجه به نظرسنجی	میزان جذابیت سیستم	**attractiveness**
بیش از 90%	نسبت تعداد نقص های علت یابی شده به تعداد کل نقص ها	میزان توانایی شناسایی علت اصلی نقص در سیستم	**analysability**
کمتر از 20%	نسبت زمان لازم برای تغییر به زمان لازم برای ایجاد بخش مورد نظر	میزان تلاش برای تغییر سیستم	**changeability**
کمتر از 8%	نسبت زمان تست به زمان ایجاد تغییر	میزان تلاش برای آزمایش یک تغییر در سیستم	**testability**
کمتر از 0.1s	response time	مدت زمان پاسخگویی	**Time behavior**
کمتر از 5GB	حافظه بر حسب GB	میزان حافظه و پایگاه داده مورد استفاده	**Resource utilisation**
کمتر از 50%	نسبت تعداد کاربرانی که بدون راهنما نصب میکنند به تعداد کل کاربران	میزان تلاش مورد نیاز برای نصب	**installability**
کمتر از max ار استاندارد ین شده	Effort و labor مورد نیاز برای تعویض	میزان سهولت تعویض یک جزء نرم افزاری معین در یک محیط مشخص	**Replaceability**

17. رویکرد برای تضمین کیفیت پروژه

برای برای تضمین کیفیت در این پروژه از روش Six Sigma استفاده می شود.

Six Sigma

این روش یکی از پر استفاده ترین استراتژی های تضمین کیفیت آماری در صنعت امروز است که بر روی کم کردن defect ها تمرکز می کند و با کاهش defect ها در مراحل انجام آن کیفیت محصول را تضمین می کند.

در این روش ضمن اینکه با جلوگیری به موقع از defect ها زمان تولید محصول را کاهش می یابد همچنین با کاهش defect هزینه های ناشی از rework کاهش می یابند.

روش شش سیگما سه قدم اصلی را تعریف می کند:

• تعریف نیازهای مشتری و اهداف پروژه و اهداف پروژه را از طریق روش های تعریف شده از ارتباطات مشتری مشخص کنید.

• اندازه گیری فرایند موجود و خروجی آن برای تعیین کیفیت فعلی (جمع آوری معیارهای defect).

• تجزیه و تحلیل معیارهای defect و تعیین عوامل بسیار حیاتی.

اگر یک فرایند نرم افزاری وجود دارد، اما بهبود لازم است، شش سیگما دو مرحله اضافی را پیشنهاد می کند:

• بهبود process با حذف علل اصلی defect .

•کنترل فرآیند تا اطمینان حاصل شود که کارهای آینده سبب ایجاد مجدد defect ها نمی شود.

این مراحل اصلی و اضافی گاهی به روش DMAIC (تعریف، اندازه گیری، تجزیه و تحلیل، بهبود و کنترل) اشاره می کند.

اگر یک سازمان در حال توسعه یک فرایند نرم افزاری ، مراحل اصلی به شرح زیر است:

• فرآیند را برای (1) جلوگیری از علل ریشه ای نقص ها و (2) رعایت الزامات مشتری، طراحی کنید.

• اطمینان حاصل کنید که مدل فرآیند، در حقیقت، از نقص ها جلوگیری می کند و نیازهای مشتری را برآورده می کند

این روش بر اساس موارد زیر شکل گرفته است:

- Prevent defects
- Reduce variation
- Focus on the customer Make decisions based on facts
- Encourage teamwork

حال به بررسی فاز های این رویکرد می پردازیم :

1. Definition phase

در این گام مساله و هدف، مشتریها ، نیازمندیهای آنان، اولویت بندی نیازمندیها و در نهایت وضعیت فرایند فعلی توصیف می شود.

فعالیت هایی که در این گام انجام می شود عبارتند از:

- Define the problem

در این قسمت مساله ی عملی و قابل دستیابی، باید بدون ابهام و به صورت کمی و طوری بیان شود که در نهایت خروجی های بدست آمده قابل اندازه گیری باشند.

بعلاوه مساله ی مطرح شده در رابطه با بهبود فرایند باید با نیازمندی های کاربر مرتبط باشد و همچنین باید مشخص شود که بهبود مورد نظر در چه مدتی حاصل می شود.

بعنوان مثال :fault tolerance سیستم 60% اندازه گیری شده و باید حداقل تا 15% ظرف مدت 3 ماه آتی بهبود یابد.

- Form a team
- Develop a project charter

Summary					
Process impact	رزرو آنلاین نوبت		تسریع روند نوبت دهی بدون ف وقت کاربران		
Team leader	مهدیس صفری				
Start date	1398/7/1		Target completion date		1398/10/1
Project description	بالا بردن fault tolerance سیستم				
benefits					
	units	current	goal	Actual hieved	Date
Sigma level		70%	85%		
Customer sat		85%	90%		
Team membership					
name	role		department	time	
مهدیس صفری	Leader		طراحی سیستم	60	
الهه رنجبری	Team member		طراحی سیستم	40	
Support required					
Support required	دسترسی به مستندات برای همه ی اعضا				
Schedule					
milestone	Target Date				
define	1398/7/10				
measure	1398/7/25				
analyze	1398/8/5				
improve	1398/9/1				
control	1398/10/1				
Critical success factor					

تعداد کاربرانی که از سیستم استفاده می کنند
تعداد عملکرد های سیستم
تعداد failure های سیستم

- Develop a project plan

در این قسمت زمانبندی گام های مختلف پروژه به همراه زمان بندی task های مرتبط با هر گام و milestoneها مشخص می شود.

- Identify the customers

با توجه به این که در تصمیمات بر اساس تاثیری که روی مشتری دارند گرفته می شوند، باید آنها را به صورت دقیق شناسایی کرد.

در این گام افرادی که با محصولات و خروجی های پروژه در ارتباط هستند مشخص می گردند:

- external (سفارش دهنده محصول)

Ultimate (در صورتی که سفارش دهنده محصول، از آن در تولید محصول دیگری استفاده کند، مشتری محصول نهایی در این دسته قرار می گیرد.)

- Internal

واحد بسته بندی (Immediate)

واحد ارسال (Intermediate)

در نهایت باید مشتریان کلیدی شناخته شوند.

برای مثال :

Customer های سیستم :

- External : بیماران ، مدیر بیمارستان ، پزشک
- Internal : ادمین اصلی

- Identify key outputs

برای مثال :

خروجی های اصلی مربوط به مشکل :

- جلب رضایت کاربر

- Identify and prioritize customer requirements

شامل پر کردن جداول زیر :

Customer: Great Auto

Requirement	Importance to Customer*	Current Satisfaction**
Order processed on time		
Order complete and accurate		
Information about products accurate		
Call answered promptly		

Customer: Packing

Requirement	Importance to Customer*	Current Satisfaction**
Order processed in time for shipping deadline		
All special requirements documented		

*Importance Ranking Scale:
1 = not very important
4 = moderately important
7 = very important
10 = extremely important
**Satisfaction Ranking Scale:
1 = not very satisfied
4 = moderately satisfied
7 = very satisfied
10 = completely satisfied

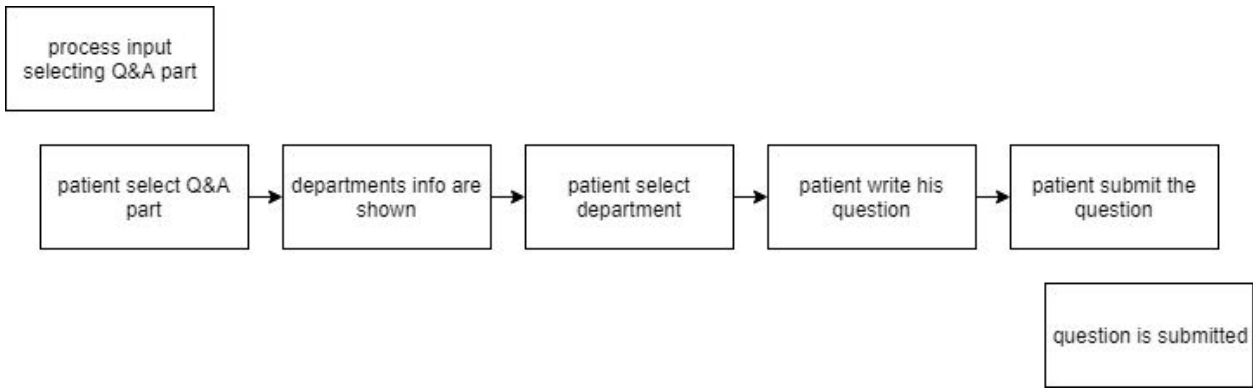
نیازمندی های کاربر سیستم (بیمار) :

نیازمندی	اهمیت	رضایتمندی اکنون
رزرو نوبت از دکتر	7	4
ارزیابی پزشک	4	7
پرسش و پاسخ	4	4

- Document the current process

از process map استفاده می شود که فرایند را به شکل دنباله ای از task ها، ورودی ها و خروجی ها نشان می دهد.

برای مثال برای پرسش بیمار:



2. Measurement phase

در این گام فرآیند فعلی و خروجی های آن مورد بررسی قرار می گیرند تا quality performance فعلی بر اساس defect metric های جمع آوری شده مشخص گردد. فعالیت هایی که در این گام انجام می شود عبارتند از:

- Determine what to measure

با توجه به اینکه پایه روش 6 سیگما براساس تحلیل آماری است، بنابراین این روش بر اندازه گیری تاکید بسیار زیادی دارد.

در رابطه با مثال علاوه بر فهم گام های مورد نیاز برای فرآیند دریافت سفارش، باید مشخص هر گام چقدر طول میکشد، چه تعداد defect در هر گام وجود دارد و تاخیرهای موجود در هر گام چقدر است تا در نهایت به نحوی از آن ها جلوگیری کرد) | و در حقیقت در اندازه گیری به دنبال معادله زیر هستیم:

$y = f(x1, x2, x3, \ldots)$

که در آن X ها مقادیر ورودی های مختلف (وابسته به گام های مختلف فرآیند)، f فرآیند و خروجی هستند.

بنابراین ابتدا باید مشخص کنیم رودیهای فرآیند چه هستند و کدام یک را باید اندازه گیری کنیم .

در وضعیت فعلی سیستم، fault tolerance 70% داریم.

sigma element 6	
Man	-
Machine	server های سیستم
Material	-
Material	-
Measurement	-
Mother nature	تعداد درخواست ها در واحد زمان

- Conduct the measurements

ا استفاده از روش مشخص شده اندازه گیری انجام می شود.

برای مثال : نسبت تعداد failure های مدیریت شده به تعداد کل failure ها در یک بازه زمانی

- Calculate current sigma level

با اندازه گیری defectها و محاسبه Dpmo و مطابقت با جدول زیر سطح سیگما را مشخص می کنیم .

DPMO	Sigma
933,193	0.00
926,471	0.05
919,243	0.10
911,492	0.15
903,199	0.20
894,350	0.25
884,930	0.30
874,928	0.35
864,334	0.40
853,141	0.45
841,345	0.50
828,944	0.55
815,940	0.60
802,338	0.65
778,145	0.70
773,372	0.75
758,036	0.80
742,154	0.85
725,747	0.90
708,840	0.95
691,462	1.00
673,645	1.05
655,422	1.10
636,831	1.15
617,911	1.20
598,706	1.25
579,260	1.30
559,618	1.35
539,828	1.40
519,939	1.45
500,000	1.50
480,061	1.55

DPMO	Sigma
460,172	1.60
440,382	1.65
420,740	1.70
401,294	1.75
382,088	1.80
363,169	1.85
344,578	1.90
326,355	1.95
308,537	2.00
291,160	2.05
274,253	2.10
257,846	2.15
241,964	2.20
226,627	2.25
211,856	2.30
197,663	2.35
184,060	2.40
171,056	2.45
158,655	2.50
146,859	2.55
135,666	2.60
125,072	2.65
115,070	2.70
105,650	2.75
96,800	2.80
88,508	2.85
80,757	2.90
73,529	2.95
66,807	3.00
60,571	3.05
54,799	3.10
49,471	3.15

DPMO	Sigma
49,471	3.15
44,565	3.20
40,059	3.25
35,930	3.30
32,157	3.35
28,717	3.40
25,588	3.45
22,750	3.50
20,182	3.55
17,865	3.60
15,778	3.65
13,904	3.70
12,225	3.75
10,724	3.80
9,387	3.85
8,198	3.90
7,143	3.95
6,210	4.00
5,386	4.05
4,661	4.10
4,024	4.15
3,467	4.20
2,980	4.25
2,555	4.30
2,186	4.35
1,866	4.40
1,589	4.45
1,350	4.50
1,144	4.55
968	4.60
816	4.65
687	4.70

DPMO	Sigma
577	4.75
483	4.80
404	4.85
337	4.90
280	4.95
233	5.00
193	5.05
159	5.10
131	5.15
108	5.20
89	5.25
72	5.30
59	5.35
48	5.40
39	5.45
32	5.50
26	5.55
21	5.60
17	5.65
13	5.70
11	5.75
9	5.80
7	5.85
5	5.90
4	5.95
3	6.00

- Determine process capability

آنچه تاکنون اندازه گیری شد، performance فرآیند بود. در ادامه باید capability نیز اندازه گیری شود.

هدف از اندازه گیری capability مقایسه Variation مشاهده در فرایند با محدودیت های تعیین شده توسط کاربر است.

voice of the process در برابر voice of the customer

- Benchmark process leaders

3. Analysis phase

برای مثال در مشکل تعداد کم پاسخگویی به کاربران به طور همزمان

در این گام داده های جمع آوری شده در گام قبل تحلیل می شود تا علت مشکلات شناسایی شده و برای آن ها راه حل پیشنهاد گردد. فعالیتهایی که در این گام انجام می شود عبارتند از:

- Determine what caused the variation

پس از یافتن علت با پرسیدن چرا، ریشه پیدا می شود و این فرایند آنقدر ادامه پیدا می کند که علت جدیدی پیدا نشود.

برای مثال : پایگاه داده توانایی پاسخگویی به درخواست همزمان را ندارد یا سرور قوی تری لازم است.

مشخص می شود مشکل از پایگاه داده است.

- Brainstorm ideas for process improvements

بعد از پیدا کردن variation ها باید برایشان راه حل پیدا کنیم.
برای مثال : باید کد پایگاه داده review شود تا بتواند به درخواست های همزمان پاسخ دهد.

- Determine which improvements would have the on meeting customer
- requirements greatest impact
- Develop a proposed process map

برای فرایند بهبود داده شده process map دیگری رسم شود.

- Assess the risks associated with the revised process

جهت تحلیل ریسک از FMEA استفاده می شود.
به کارگیری آن به صورت زیر است:

- مشخص نمودن حالاتی که در آن فرآیند ممکن است با شکست مواجه شود.
- مشخص نمودن میزان تاثیر (effect) این حالت شکست
- مشخص نمودن روش های جلوگیری از بروز این حالت شکست
- تهیه یک action plan برای جلوگیری از بروز حالت شکست و مستندسازی نتایج آن

4. Improvement phase

در این گام براساس نتایج به دست آمده از گام های قبل، تغییراتی که باید در نهایت اعمال شوند مشخص می گردند و بهبود فرآیند انجام می پذیرد.
فعالیت هایی که در این گام انجام می شود عبارتند از:

- Gain approval for the proposed changes

گزینه های پیشنهادی بر اساس هزینه، تاثیز و ریسکهای مرتبط مررسی شده و گزینه مناسب انتخاب می شود.

- Finalize the implementation plan
- Implement the approved changes

برای مثال :
یکی از راه حل های ممکن اختصاص فضای بیشتر به پایگاه داده و دیگر اینکه اگر پایگاه داده به صورت SQL است از NO SQL استفاده شود و پس از انتخاب راه حل مناسب راه حل نهایی را پیاده سازی کرده و میزان پیشرفت را گزارش می دهیم.

5. Control phase

Six sigma پس از بهبود فرآیند متوقف نمی شود، بلکه همواره به دنبال تضمین این است که بهبود به دست آمده از دست نخواهد رفت. به این ترتیب تغییرات اعمال شده و سود به دست آمده همیشگی خواهند بود.
فعالیت هایی که در این گام انجام می شود عبارتند از:

- Establish key metrics

باید یک سری متریک های reliable تعریف کرد.

برای نمونه :

Data Reliability	Quantify the objectivity of the data: 1 = subjective, no historical basis 4 = based on anecdotal historical experience 7 = based on direct observation 10 = obtained directly from an objective source (e.g.,computer system, time stamp)

- Develop the control strategy

متریک ها برای تضمین حفظ بهبود بدست آمده باید یک استراتژی برای کنترل داشته باشند که در قالب یک برنامه مطابق نمونه های اسلاید تهیه شود.

- Implement the control plan

برنامه ارایه شده باید مانیتور شود و بر اساس آن گزارش های لازم تولید شود.

- Measure and communicate improvements

گزارش های ماهیانه در قالب نمودارهایی که نمونه ی آنها در اسلایدها آمده ارایه می شوند که براساس میزان و روند بهبود در ماه های مختلف قابل ملاحظه است.

به طور کلی برای مثال با تست های متعدد از بهبود عملکرد پایگاه داده اطمینان حاصل می کنیم و اگر لازم باشد بخاطر تغییرات اعمال شده قسمت های دیگر را نیز تغییر دهیم با کنترل کیفیت بخش هایی که نیاز به تغییر دارند تغییرات را اعمال می کنیم و با ایجاد مستندات لازم را ایجاد می کنیم رسیدن به سطح حداقل 4 مراحل را تکرار می کنیم.

SQA plan .18

.

طبق تعریف کتاب طرح SQA یک نقشه راه برای ایجاد تضمین کیفیت نرم افزار فراهم می کند. این طرح توسط گروه SQA (یا تیم نرم افزاری اگر گروه SQA وجود ندارد)،به عنوان یک الگو برای فعالیت های SQA است که برای هر پروژه نرم افزاری ایجاد شده است.

در ادامه طبق استاندارد IEEE standard Std 730-1989 برنامه ی تضمین کیفیت این پروژه را تعریف می کنیم.

1. هدف :

هدف از تولید این سند، مشخص کردن یک الگو برای فعالیت های SQA مانند روند process ها ، استانداردها و ابزار ها و تکنیک های مورد نیاز در پروژه برای تضمین کیفیت محصول پروژه است.
آنچه در روند این برنامه تضمین کیفیت مشخص می شود :

- معرفی تیم QA و مشخص کردن وظایف هر یک از اعضای تیم
- مشخص شدن فرایند های بررسی تضمین کیفیت
- مشخص شدن خروجی هر مرحله
- روش REVIEW پروژه

2. مراجع :

- Software Engineering, Roger S. Pressman
- IEEE Standard Std 730-1989

3. مدیریت :

:Organization
استاد: دکترعبدالله زاده بارفروش
Developer: ...

اعضای تیم :
مدیریت تضمین کیفیت : مهدیس صفری
برنامه نویسان : ...

وظایف :

- تهیه کردن requirement specification و cost estimation
- تهیه کردن design plan و test plan
- پیاده سازی تست ها
- تیم توسعه دهنده پس از اتمام تجزیه و تحلیل، مرحله طراحی و آزمایش، گزارش رسمی ای ارائه می دهد که با review آن ها بازخورد هایی ایجاد می شود.

4. مستندات :

اسناد زیر در پایان هر مرحله ارائه می شود.

فاز 1: نیازمندی های نرم افزاری

- Project Overview
- System Diagram
- Project Gantt Chart Timeline
- Cost Analysis
- (System Requirements Specification (SRS

مرحله 2: طراحی نرم افزار

- برنامه تضمین کیفیت نرم افزار (SQAP)
- Formal Specifications
- Test Plan
- (Formal Technical Review (checklist

مرحله 3: اجرای نرم افزار

- راهنمای کاربری
- تست و ارزیابی قابل اطمینان
- ارزیابی پروژه

ضمیمه:

- Source Code

5. استاندارد ها و metric های مورد نظر تیم تضمین کیفیت :

- استاندارد ها :

  استاندارد document نویسی : python documentation

  استاندارد کد نویسی : python 3

  استاندارد تضمین کیفیت : six sigma

  استاندارد تست ها : IEEE Standard for Software Test Documentation

- Metrics :

  LOC

6. Review :

سه formal presentation توسط توسعه دهنده تهیه شده و توسط کمیته در پایان هر مرحله (نیازمندی، طراحی و پیاده سازی) مورد ارزیابی قرار خواهد گرفت.
در بررسی اولیه طراحی در پایان مرحله طراحی، کفایت تکنیک طراحی سطح بالا ارزیابی خواهد شد. تمام اسناد مربوط به یک فاز خاص در بررسی ارائه خواهد شد. ارزیابی باید توسط اعضای کمیته تصویب شود. تمام کمبودها یا ناسازگاری باید توسط توسعه دهنده تصحیح شود و مجددا به اعضای کمیته برای تأیید مجدد ارسال شود.

7. Test plans , tools , techniques :

یک برنامه تست نرم افزاری (STP) برای پاسخگویی به نیازهای نرم افزاری نوشته شده است. این طرح مدیریت و تست عملکرد را با یک مرور کلی از فعالیت های تست، برنامه ها و منابع مورد نیاز برای انجام آزمایش فراهم می کند. این طرح توضیح می دهد که چگونه مشخصات آزمایشگاهی در بخش های زیر اجرا می شود.
واحد آزمایش:
تمامی کد ها مورد آزمایش قرار می گیرند تا اطمینان حاصل شود که واحد واحد (کلاس) عملکردهای مورد نیاز را انجام می دهد و نتایج و داده های مناسب را خروجی می دهد.
آزمون یکپارچه سازی:
دو سطح آزمون یکپارچه وجود دارد. یک سطح فرایند تست قابلیت نرم افزار است. سطح دوم تست ادغام زمانی اتفاق می افتد که ماژول های کافی برای نشان دادن سناریو یکپارچه شده اند.
برای اندازه گیری سایز پروژه و آنالیز های لازم نیز ابزار COCOMO II و برای رمزگذاری ارتباط بین سرور و سرویس گیرنده از چارچوب Cryptix استفاده می شود.

8. اعلام مشکل و رفع مشکلات :

در طول پروژه اعضای تیم توسعه دهنده می تواند مشکلات را به گروه تضمین کیفیت گزارش دهد. گروه تضمین کیفیت پیشنهادات خود را ارائه می دهد.
گروه تضمین کیفیت همچنین می تواند سؤالاتی را مطرح کند و به آنها توصیه کند و در نهایت توسعه دهنده اشتباهات خود را تصحیح می کند.

9. کنترل کردن کد :

N/A

10. کنترل کردن media :

N/A

11. Supplier کنترل :

N/A

12. Records :

تمامی مستندات حاصل در وبسایت مربوط به پروژه های مدیر پروژه ثبت می شود.

13. Training :

دانش مورد نیاز برای تیم تولید :

- برای مثال یکی از این دانش ها آشنایی کامل با ابزار Visual Paradigm است

دانش مورد نیاز تیم تضمین و کنترل کیفیت :

- آشنایی کامل با ابزار Six sigma
- آشنایی کامل با ابزار II COCOMO
- آشنایی با ابزار audit

منبع برنامه تضمین کیفیت :
http://people.cs.ksu.edu/~yli3568/mse/SQA-032802.doc

19. تکنیک های کنترل کیفیت

1. Six sigma

همانطور که در بخش تضمین کیفیت دیدیم این روش طی مراحل معینی، مشکل تحلیل شده و رویکرد مناسب برای حل آن انتخاب شده و میزان بهبود سیستم پس از اجرای راه حل اندازه گیری می شود.
با استفاده از این روش در طول فرایند تولید پروژه می توان کیفیت سیستم را تضمین کرد.

2. Software inspection

inspection یکی از رایج ترین شیوه های بازبینی در پروژه های نرم افزاری است.
هدف inspection این است که defect ها شناسایی شوند.
معمولا محصولات مورد بازرسی(inspection) شامل مشخصات نیازمندی های نرم افزار و برنامه های تست است.
در inspection، یک محصول کار برای review انتخاب شده و یک تیم برای جلسه inspection برای review محصول کار جمع آوری شده است.
یک moderator برای نظارت بر جلسه انتخاب شده است.
هر inspector با خواندن work product و note کردن defect های سیستم برای meeting آماده می شود.

در طی بازرسی، نقش های زیر استفاده می شود :

- author : فردی که محصول مورد بازرسی را را ایجاد کرده است.

- moderator : رهبر بازرسی است و برنامه ریزی و هماهنگی بازرسی را بر عهده دارد.
- reader: شخصی که از طریق مستندات یک مورد را در یک زمان می خواند. بازرسان دیگر سپس defect را نقد می کنند.
- Recorder / Scribe: فردی که defect هایی را که در طی بازرسی پیدا می شود، مستند می کند.
- inspector : فردی که work product را بررسی می کند تا نقایص احتمالی را شناسایی کند.

مراحل inspection process به صورت زیر است :

- Planning

بازرسی توسط moderator برنامه ریزی، و هدف، زمان بندی و شرکت کنندگان هر meeting مشخص می شود.

- Overview meeting

author زمینه work product را توصیف می کند.

- Preparation

هر inspector محصول را برای شناسایی defect های احتمالی بررسی می کند.

- Inspection meeting

در طول این جلسه، خواننده work product را بخش به بخش می خواند و بازرسان نقص ها را برای هر بخش مشخص می کنند.

- Rework

نویسنده با توجه به برنامه های عملی جلسه بازرسی، تغییرات را به محصول کاری اعمال می کند.

- Follow-up

تغییرات توسط author بررسی می شود تا مطمئن شود همه چیز صحت دارد.

با انجام inspection از ابتدا و در طول فرایند انجام پروژه می توان با مشارکت همه ی اعضای تیم artifact های هر مرحله را تحلیل و کیفیت آن ها را تضمین کرد.

https://en.wikipedia.org/wiki/Software_inspection

20. متریک ها technical review و بررسی از نظر اقتصادی

طبق اسلاید های درس متریک ها technical review را میتوان به این صورت تعریف کرد :

- Preparation effort یا E_p

  تلاش مورد نیاز برای review یک محصول کار قبل از جلسه review واقعی

  در Planning & preparation در مدل مرجع محاسبه می شود.

- Assessment effort یا E_a

  تلاشی که در روند review واقعی محصول صرف می شود

  در Meeting structure در مدل مرجع محاسبه می شود.

- rework effort یا E_r

  میزان فعالیت لازم برای تصحیح خطاهای پوشش داده نشده در جلسه review

  در Correction & verification و Roles Individuals Play در مدل مرجع محاسبه می شود.

- Major error found یا Err_{major}

  تعداد خطاهای major (نیازمند تلاش کمتر برای تصحیح) پیدا شده

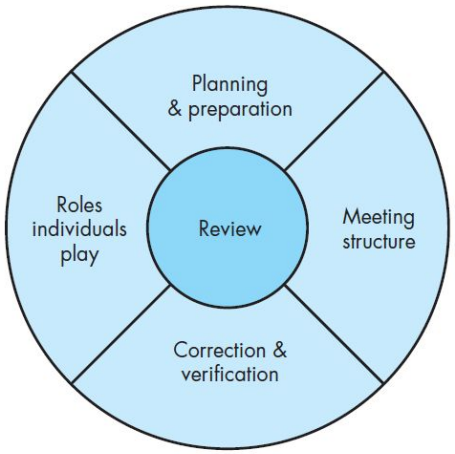
- Minor error found یا Err_{minor}

تعدا خطاهای minor (نیازمند تلاش کمتر برای تصحیح) پیدا شده

- Work product size یا wps

سایز محصولی که مورد بررسی (به عنوان مثال تعداد مدلهای UML یا تعداد صفحات سند یا تعداد خطوط کد)

مدل مرجع :



که total review effort و total number of errors از روابط زیر بدست می آیند.

$E_{review} = E_p + E_a + E_r$

$E_{rrtotal} = E_{rrminor} + E_{rrmajor}$

Error Density= $E_{rrtotal}$ / WPS

Effort saved per error = Etesting – Ereviews

متریک	واحد	مقدار
Ep	نفر-ساعت	50
Ea	نفر-ساعت	70
Er	نفر-ساعت	120
WPS	LOC	5000
Err_{minor}	تعداد	60
Err_{maior}	تعداد	30

$Ereview = Ep + Ea + Er = 240$

$Error_{total} = E_{major} + E_{minor} = 90$

Error Density= $E_{rrtotal}$ / WPS = 90 / 5000 = 0.0018 -> 0.02

در بررسی داده های review جمع آوری شده، می بینید که خطاهای جزئی حدود 6 برابر بیشتر از خطاهای عمده رخ می دهد.

the average effort to find and correct a requirements error during review -> Ereview

Ereview = (6*60 + 30)/7 = 56 p-h

$E_{testing}$ = 90 p-h : میانگین تلاش لازم برای پیدا کردن خطا در مرحله ی تست

Effort saved per error = Etesting – Ereviews = 90 - 56 = 34 p-h

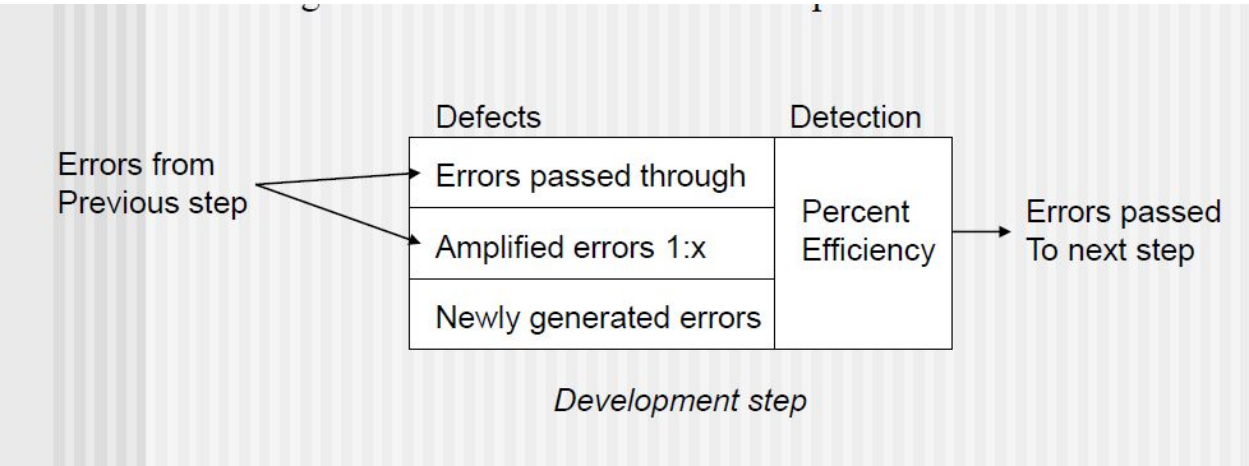
از آنجایی که 240 خطا در بازیابی پیدا شده، هزینه ای که در هر review ذخیره می شود :

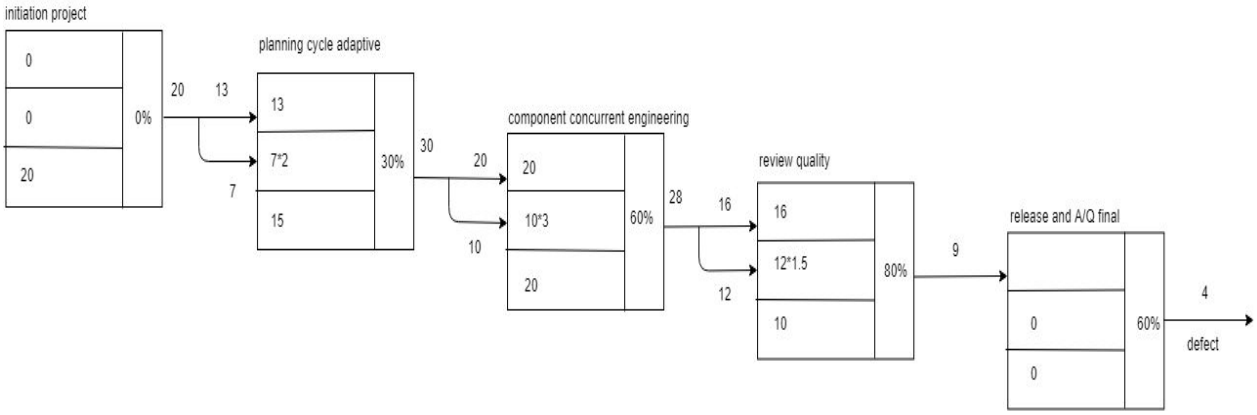
240*34 = 8,160 p-h

21. مدل defect amplification

همانطور که در اسلاید ها آمده یک مدل defect amplification برای نشان دادن تولید و شناسایی خطاها در طی اقدامات طراحی و تولید کد یک فرآیند نرم افزاری استفاده می شود.

در ادامه بر اساس تعریف زیر مدل defect amplification خود را رسم می کنیم :





22. چرخه حیات،استراتژی و واحد تست

استراتژی تست باید توضیح دهد که چگونه و چه زمانی آزمایش انجام خواهد شد، چه کسانی آزمایش را انجام خواهند داد، نوع تست انجام شده، ویژگی های آزمایش شده، محیط (ها) که در آن تست انجام می شود، چه ابزار تست استفاده می شود، و چگونه نقص ردیابی و مدیریت استراتژی تست باید توسط تیم هسته چابک تهیه شود.

- Test Actors

برای موثر بودن تست حقوق مختلف اختصاص یافته به گروه های امنیتی مختلف، تیم باید یک سازمان تست کوچک را با پیگیری این مراحل ایجاد کند:

- شناسایی و ایجاد Actor های مختلف که حقوق دسترسی CA PPM باید مورد تایید قرار گیرد. معمولا این یک بازیگر برای هر گروه امنیتی CA PPM است.
- اگر یک سیستم سازمانی گزارش (OBS) در سیستم استفاده شود، ارتباط مناسب گزارشی را برای Actor های تست ایجاد کنید.
- برای دسترسی به هر یک از گروه های امنیتی، با وارد شدن به عنوان Actor تست و اجرای هر مورد آزمون، مدارک دسترسی CA PPM را تأیید کنید.

- Types of Testing

انواع مختلف تست در نقاط مختلف در طول چرخه عمر پیاده سازی پروژه رخ می دهد. هر نوع تست هدف خاصی را دنبال می کند، اما در نهایت همه final user acceptance را قبل از نهایی شدن سیستم انجام می دهند.

- Testing Environments

هرگاه تغییرات از یک محیط به محیط دیگر منتقل شود، توصیه می شود که تست های انجام شده در محیط قبلی در محیط جدید را مجددا تایید کنید.

- Test Automation and Testing Tools

این باعث کاهش احتمال خطاهای انسانی در حین آزمون رگرسیون و آزمون های تکراری می شود. تیم پروژه باید مدیریت تست و ابزارهای اتوماسیون مورد نیاز برای اجرای تست را تعریف کند. اگر هیچ ابزار تست خودکار در دسترس نیست، test case ها و نتایج تست باید به طور جداگانه، مانند یک spreadsheet، ردیابی شوند.

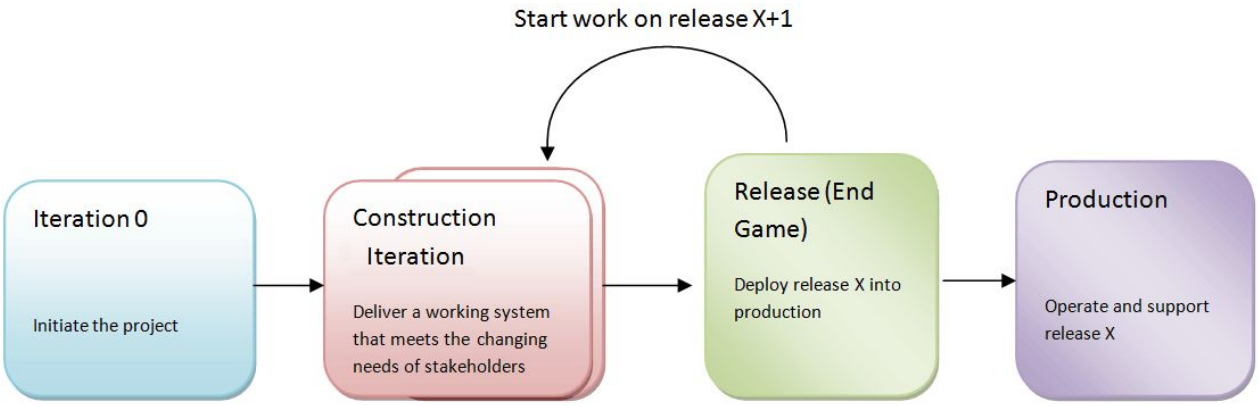
- Risk Analysis

تیم اصلی چابک باید یک لیست جامع از تمام خطرات بالقوه ایجاد کند. این خطرات باید بر اساس احتمال و تاثیر بالقوه اولویت بندی شوند. این اطلاعات می تواند توسط تیم پروژه های چابک مورد استفاده قرار گیرد تا تمرکز بر برنامه هایی برای کاهش این خطرات و ایجاد شرایط احتمالی باشد. درست مانند هر پروژه دیگر، اگر یک روش مدیریت ریسک به راحتی در دسترس باشد، باید آن را استفاده کرد.

- Test Planning and Execution
  - استفاده از cases/user story ها به وضوح تعریف شده اند.
  - ساخت test case ها و test script ها بر اساس موارد کاربرد.
  - تعریف معیارهای پذیرش برای هر ویژگی و هر فرآیند.
  - انجام تست.
  - بررسی و تایید نتایج آزمون.
  - تغییرات را به تولید یا برای سطح بعدی تست (مانند user acceptance testing) منتقل کنید.
- Review and Approval
  - تیم چابک اصلی و رهبر پروژه باید مسئول اعتبار و تأیید تمام نتایج آزمون باشند.
  - مدیران عملیاتی سیستم های رابط نیز باید نتایج integration test را امضا کنند.
  - کاربران نهایی و نمایندگان مجاز آنها باید نتایج User Acceptance Test را تایید و امضا کنند.

https://www.ca.com/content/dam/ca/us/files/services-brief/ca-ppm-leading-practice-test-strategy-and-approach-in-agile-projects.pdf

چرخه عمر تست Agile شامل چهار مرحله می شود



Agile Testing Strategy

- **Iteration 0**

در این مرحله، انجام وظایف تنظیمات اولیه را انجام می دهیم که شامل شناسایی افراد برای آزمایش، نصب ابزارهای تست، برنامه ریزی منابع و ... است.

مراحل زیر برای دستیابی به Iteration 0 تنظیم شده است :

- ایجاد یک مورد کسب و کار برای این پروژه
- ایجاد شرایط مرزی و محدوده پروژه
- نیازمندی های کلیدی را در نظر بگیرید و از موارد کاربردی استفاده کنید که مشکلات طراحی را رفع می کنند.
- یک یا چند معماری نمونه را شرح دهید
- شناسایی خطر
- برآورد هزینه و تهیه یک پروژه اولیه

- **Construction Iterations**

اکثر آزمایش ها در این مرحله رخ می دهد. در مرحله، تیم چابک نیازمندی ها را طبق اولویت تعیین شده پیاده سازی می کند.

این مرحله شامل confirmatory testing و investigative testing است.

confirmatory testing تمرکز بر سیستم اهداف ذینفعان را همانطور که گفته شده برآورده می کند و توسط تیم انجام می شود. در حالی که investigative testing مشکلاتی را که تیم confirmatory آن را نادیده گرفته است حل می کند .

در investigative testing، مشکلات بالقوه را در قالب defect story ها تعیین می شوند.

investigative testing با مسائل رایج مانند integration testing, load/stress testing و security testing انجام می شود.

برای confirmatory testing دو جنبه developer testing و agile acceptance testing وجود دارد. هر دو آنها به صورت اتوماتیک برای تست رگرسیون پیوسته در طول چرخه حیات فعال هستند. Confirmatory testing معادل چابکی برای تست کردن مشخصات (specification) است.
Agile acceptance testing ترکیبی از تست functional و تست acceptance قدیمی است که تیم توسعه و ذینفعان آن را با هم انجام می دهند. در حالی که developer testing ترکیبی از unit testing و service integration testing قدیمی است.
Developer testing کد برنامه و طرح پایگاه داده را بررسی می کند.

- Release End Game Or Transition Phase

هدف این مرحله این است که سیستم را به طور موفقیت آمیز منتشر کنید. فعالیت ها در این مرحله شامل آموزش افراد نهایی، حمایت از مردم و افراد عملیاتی است. همچنین شامل بازاریابی انتشار محصول، پشتیبان گیری و بازسازی، نهایی شدن سیستم و اسناد کاربر است.
مرحله تست نهایی شامل تست کامل سیستم و acceptance test است. برای اینکه مرحله تست نهایی خود را بدون هیچ گونه موانعی به اتمام رسانیم، باید آن را در مرحله ی construction iterations دقیق تر آزمایش کنیم. در End Game، تسترها بر روی defect story ها کار خواهند کرد.

- Production

بعد از مرحله release، محصول به مرحله تولید منتقل می شود.

تست	هدف تست	واحد تست	ورودی تست	خروجی تست
confirmatory testing	تایید برآورده شدن اهداف ذینفعان سیستم	Use case	Test plan + مدل نیازمندی	نیازمندی های پوشش داده نشده + عملیات های بدون نیازمندی
developer testing	تأیید عملکرد صحیح	کلاس ها و پایگاه داده	Test plan + کد برنامه و طرح پایگاه داده	خطاهای تعریف کلاس ها یا اشکالات پیاده سازی
agile acceptance testing	تأیید اینکه سیستم عملکردهای آن را به عهده دارد	Use case	Test plan + مدل نیازمندی + Use case	نیازمندی های پوشش داده نشده + عملیات های بدون نیازمندی
investigative testing	مشکلاتی را که تیم confirmatory آن را نادیده گرفته است حل می کند	سیستم	Test plan + کلاس های برنامه (تعریف یا کد)	گزارش نحوه عملکرد سیستم در شرایط مختلف
load/stress testing	اطمینان از بازیابی سیستم پس از fail	سیستم	Test plan + کلاس های برنامه (کد)	گزارش نحوه عملکرد سیستم در شرایط مختلف

integration testing	درستی عملکرد اجزای سیستم در ارتباط با هم	Build	+ Test plan کلاس های برنامه (تعریف یا کد)	خطاهای منطقی در ارتباط بین کلاس ها یا خطا در عدم همخوانی interface و یا ارورهای عملیاتی
security testing	تایید امنیت سیستم	سیستم	+ Test plan کلاس های برنامه (کد)	گزارش نحوه عملکرد سیستم در شرایط مختلف

https://www.guru99.com/agile-testing-a-beginner-s-guide.html

23. برنامه ی تست بر اساس w5h2 question

integration testing :

پرسش	پاسخ
What	یکی از انواع تست که روابط بین کلاس ها را بررسی می کند. (بررسی اینکه interface و scale خروجی های هر دو کلاس مرتبط با هم ، بای کدیگر همخوانی داشته باشند)
Why	برای اطمینان از درستی کارکرد کلاس ها با یکدیگر
Who	• test leader • test manager • test designer • tester
Where	روی build های مشخصی از هر فاز در پروژه
When	در طی فرایند پروژه و بعد از پایان هر فاز
How	برای هر سناریو به صورت سلسله مراتبی ارتباطات هر کلاس را با کلاس های مرتبط بررسی میکنیم.ابزار مورد استفاده برای این تست در این پروژه Worksoft است.
How much	تا زمانی که تمام سناریو های ممکن چک شوند و به درجه کیفیت مورد نظر برسد در عین حال از نظر زمانی و هزینه صرف داشته باشد.

Scenario based testing .24

تست سناریو یک تکنیک تست نرم افزاری است که انجام می شود تا اطمینان حاصل شود که کارکرد نرم افزار در حال کار است یا تمام روند جریان کار درست است. در آزمایش سناریو، تسترها خود را در کفش های کاربران نهایی قرار می دهند و از مواردی استفاده می کنند که میتواند توسط کاربر نهایی توسط نرم افزار انجام شود.

در آزمایش سناریو، آزمایش کنندگان از مشتریان، ذینفعان و توسعه دهندگان برای ایجاد سناریوهای آزمایش استفاده می کنند.

تست سناریو به آزمایشکنندگان کمک می کند تا کشف کنند که چگونه نرم افزار در دست یک کاربر نهایی کار می کند. از آنجا که تست سناریو جریان تجارت نرم افزار را آزمایش می کند، آن را در یافتن بسیاری از نقص هایی که نمی توانند با سایر انواع آزمون ها یافت شوند کمک می کند.

در ادامه سه سناریو از سناریو های سیستم برای نمونه آورده شده است.

سناریو 1 :
کاربر در نقش بیمار log in می کند و وارد صفحه ی ارزیابی پزشک می شود. در این صفحه بخش های مختلف بیمارستان به کاربر نمایش داده می شود که باید یکی را انتخاب کند سپس وارد صفحه ی انتخاب پزشک مورد نظر می شود و ارزیابی خود را وارد کرده و ثبت می کند.
مواردی که در این سناریو باید چک شوند :

1. سیستم احراز هویت برای ورود کاربر به درستی کار کند.
2. کاربر پس از ورود، وارد صفحه ی انتخاب بخش شود.
3. با انتخاب صفحه هر بخش اطلاعات به درستی نمایش داده شوند.
4. با انتخاب صفحه هر پزشک اطلاعات به درستی نمایش داده شوند.
5. امکان مشاهده ی ارزیابی ها دیگر کاربران باشد.
6. امکان درج ارزیابی باشد.
7. ثبت ارزیابی به درستی انجام شود.
8. در صورت مواجهه با خطا در ثبت، پیغام خطا به کاربر نمایش داده شود.

سناریو 2 :
کاربر در نقش پزشک log in می کند و وارد صفحه ی خود می شود. در این صفحه پزشک میتواند ارزیابی های خود را مشاهده کند.
مواردی که در این سناریو باید چک شوند :

1. سیستم احراز هویت برای ورود کاربر به درستی کار کند.
2. کاربر پس از ورود، وارد صفحه ی خود شود.
3. امکان مشاهده ی ارزیابی های کاربران باشد.
4. امکان درج ارزیابی نباشد.
5. در صورت مواجهه با خطا در ثبت، پیغام خطا به کاربر نمایش داده شود.

سناریو 3 :
کاربر در نقش بیمار log in می کند و وارد صفحه ی رزرو نوبت می شود. در این صفحه بخش های مختلف بیمارستان به کاربر نمایش داده می شود که باید یکی را انتخاب کند سپس وارد صفحه ی انتخاب پزشک مورد نظر می شود و ارزیابی خود را وارد کرده و ثبت می کند.

1. سیستم احراز هویت برای ورود کاربر به درستی کار کند.
2. کاربر پس از ورود، وارد صفحه ی انتخاب بخش شود.
3. با انتخاب صفحه هر بخش اطلاعات به درستی نمایش داده شوند.
4. با انتخاب صفحه هر پزشک اطلاعات به درستی نمایش داده شوند.
5. برنامه پزشک به درستی نمایش داده شوند. (نوبت هایی که امکان رزرو دارند نمایش داده شوند.)
6. امکان کلیک بر روی هر نوبت وجود داشته باشد.
7. امکان ثبت نوبت وجود داشته باشد.
8. ثبت نوبت به درستی انجام شود و پیامی برای تایید به کاربر نمایش داده شود.
9. در صورت مواجهه با خطا در ثبت، پیغام خطا به کاربر نمایش داده شود.

25. CRC 5 برای سیستم

یک مدل CRC در واقع مجموعه ای از کارت های شاخص استاندارد برای نمایش گرافیکی از class diagram و نمایش روابط بین کلاس هاست. کارت ها به سه بخش تقسیم می شوند. در بالای کارت شما نام کلاس را می نویسید. در سمت چپ کارت مسئولیت های کلاس و همکاران در سمت راست لیست می شود.

Class name : Doctor	
Responsibilities	**Collaborators**
answer	qans answer
display_qn	qans
display_assessment	doctors assessment
display_sch	doctors schdule
display_medrec	patient history

Class name : patient	
Responsibilities	**Collaborators**
ask	qans question
reserve	Reservation_info reservation
assess	doctors assessment
display_qn	qans

display_assessment	doctors assessment

Class name : web manager	
Responsibilities	**Collaborators**
new_doc	Users doctor
new_sch	doctors schdule
remove_doc	Users doctor

Class name : qans	
Responsibilities	**Collaborators**
insert_reply	answer doctor
insert_question	Patient question
display	Department Patient doctor

Class name : patient history	
Responsibilities	**Collaborators**
modify	Patient doctor
modify	Patient doctor

26. تمرین 24.7 کتاب

طبق توضیحات کتاب پرسمن :

1. برای هر client class، از لیست عملیات کلاس استفاده کنید تا یک سری test sequences تصادفی تولید شود. عملیات پیام ها را به سایر کلاس های سرور ارسال می کند.

2. برای هر پیام تولید شده، کلاس مشترک و عملیات متناظر را در شیء سرور در نظر بگیرید.

3. برای هر عملیات در شیء سرور (که توسط پیام های فرستاده شده از شیء مشتری فراخوانی شده است)، پیام هایی را که می فرستد در نظر بگیرید.

4. برای هر یک از پیام ها، سطح بعدی عملیاتی که فراخوانی می شود را در نظر بگیرید و این ها را در توالی تست قرار دهید.

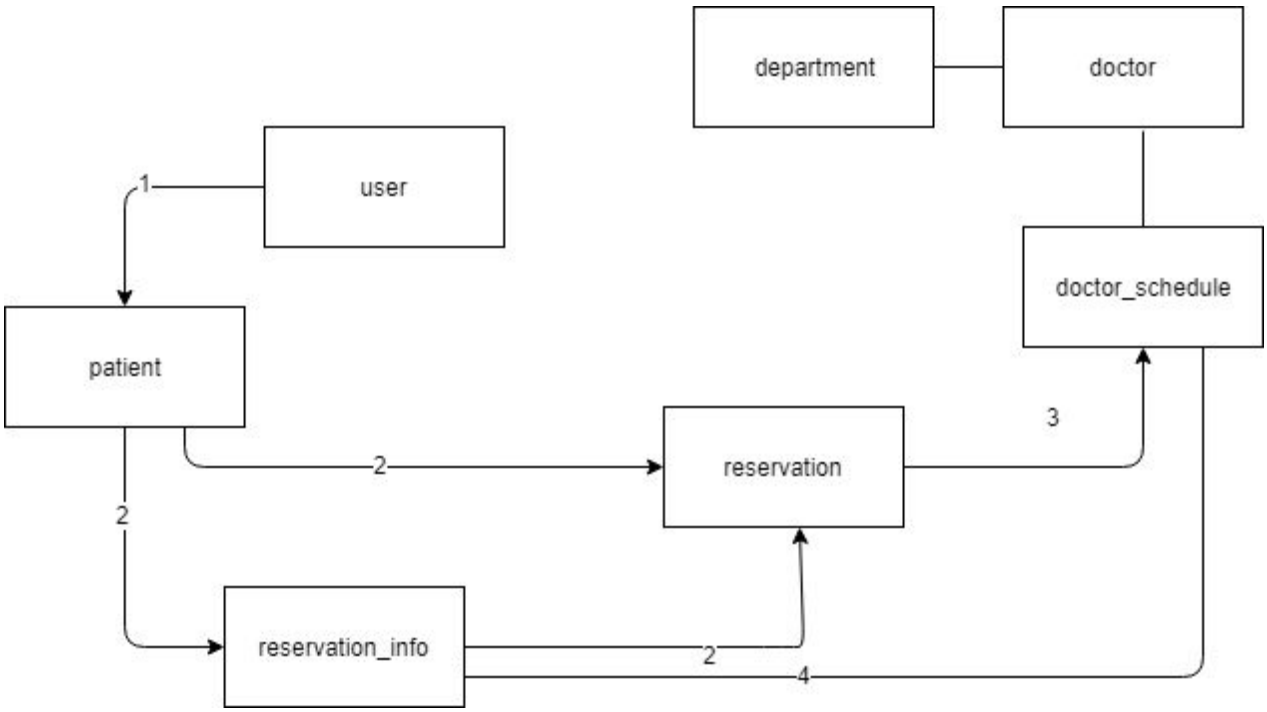
برای این تست سناریو هایی را که با استفاده از چند کلاس مرتبط بهم ایجاد می شوند را در نظر میگیریم.

در اینجا یکی از سناریو ها را مثال می زنیم.

Testcase1 :

کاربر در نقش بیمار log in می کند و وارد صفحه ی رزرو نوبت می شود. در این صفحه بخش های مختلف بیمارستان به کاربر نمایش داده می شود که باید یکی را انتخاب کند سپس وارد صفحه ی انتخاب پزشک مورد نظر می شود و ارزیابی خود را وارد کرده و ثبت می کند.

login(username, pass) - reserve(department,doctor,reservation_info) - modify_schedule - update_schedule



برای این کار ابتدا کاربر در کلاس user احراز هویت می شود و به عنوان بیمار شناخته می شود سپس کلاس reservation اطلاعات لازم برای رزرو نوبت ، یعنی بخش، پزشک و اطلاعات نوبت مورد نظر بیمار را از طریق تابع reserve که در کلاس patient صدا زده شده دریافت می کند. اطلاعات نوبت مورد نظر در قالب کلاس reservation_info در دسترس کلاس reservation قرار می گیرد. کلاس reservation با تابع modify_schedule ، کلاس doctor_schedule را

به روز رسانی می کند که برای این کار کلاس doctor_schedule از تابع update_schedule و کلاس reservation_info استفاده می کند.

به این شکل روابط این کلاس های مرتبط با هم با استفاده از توابع مورد استفاده، بررسی می شود.

Gantt chart .27

در پیوست آمده است.

28. متریک ها در گام های مختلف چرخه ی حیات به همراه , measurement measure

❑ initiation project

Measurement	measure	metric
4	عدد	تعداد actor ها
17	عدد	تعداد usecase ها
50	نفر-ساعت	Effort
20	عدد	تعداد اعضای درگیر در این مرحله

❑ planning cycle adaptive

Measurement	measure	metric
25	عدد	تعداد component ها
3	عدد	میانگین تعداد function های هر class
4	عدد	تعداد iteration
80	نفر-ساعت	Effort در این مرحله
25	عدد	تعداد اعضای درگیر در این مرحله
6	عدد	میانگین تعداد component های هر iteration

❑ component concurrent engineering

Measurement	measure	metric
90000	عدد	تعداد خط کد
150	عدد	تعداد module ها
10	عدد	تعداد مدل های طراحی شده
500	p-h	Effort پیاده سازی
200	p-h	Effort در rework

23	عدد	تعداد اعضای درگیر در این مرحله

❑ review quality

Measurement	measure	metric
66718	عدد	تعداد خط کد
1000	عدد	خط مستندات ایجاد شده
90	عدد	تعداد test case های طراحی شده
45	p-h	Effort تست
30	p-h	Effort تحلیل
250	p-h	Effort در rework در این مرحله
20	عدد	تعداد اعضای درگیر در این مرحله
15	عدد	تعداد defect ها

❑ release and A/Q final

Measurement	measure	metric
20	p-h	Effort ارزیابی سیستم
100	عدد	خط مستندات نهایی ایجاد شده
7	عدد	تعداد اعضای درگیر در این مرحله

29. محاسبه ی function point در سیستم

مطابق کتاب function point از رابطه ی زیر بدست می آید:

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

که (i = 1 to 14) F i در واقع value adjustment factors (VAF) هستند که با پاسخ به سوالات زیر مشخص می شوند :

آیا سیستم به پشتیبانی و قابلیت بازیابی قابل اعتماد نیاز دارد؟	5
آیا برای انتقال اطلاعات از/به برنامه نیاز به ارتباطات داده ای مشخصی است؟	2
آیا تابع ها و عملیات پردازشی توزیع شده وجود دارد؟	1
آیا performance اهمیت بالایی دارد؟	5

آیا این سیستم در یک محیط عملیاتی موجود و به شدت مورد استفاده اجرا می شود؟	3
آیا سیستم به ورودی داده ای آنلاین نیاز دارد؟	5
آیا ورود اطلاعات آنلاین به تراکنش ورودی نیاز دارد تا روی عملیات مختلف ساخته د؟	3
آیا ILF ها بصورت آنلاین بروزرسانی می شود؟	4
آیا ورودی ها، خروجی ها یا پرس و جو ها پیچیده هستند؟	2
آیا پردازش داخلی پیچیده می باشد؟	1
آیا کد برای استفاده مجدد طراحی شده است؟	2
آیا تغییر و نصب در طراحی در نظر گرفته شده است؟	2
آیا سیستم برای نصب های متعدد در تشکیلات مختلف طراحی شده است؟	4
آیا سیستم طوری هست که کاربر به راحتی از آن استفاده کند و تغییرات را تسهیل ؟	4
مجموع	43

هر یک از این سوالات با استفاده از یک مقیاس مرتبه ای که از 0 (مهم نیست یا قابل اجرا) تا 5 (کاملا ضروری) است پاسخ داده شده اند.

حال برای محاسبه ی count total جدول زیر را پر می کنیم.

Information domain val	count	weight			total
		simple	average	complex	
External inputs	12	2	5	6	24
External outputs	5	5	6	7	25
External inqueries	12	4	6	7	48
Internal logic file	4	6	8	11	24
External interface file	2	4	6	10	8
Total = 129					

FP = 129* [0.65 + 0.01 * 43] = 171

33. تکنیک تخمین پروژه

دو روش متداول برای تخمین پروژه، تعداد خطوط (LOC) و FP می باشد.
همانطور که در بخش 29 مشاهده کردیم ، FP سیستم برابر 171 بود.

تعداد خطوط برای بخش های مختلف :

Login : 300
Logout : 250
Signup : 200
Patient class : 9500
Doctor class : 8000
Hospital manager class: 4000
Web manager class : 15000
Database : 7000
Interface : 5000
Help : 1000
UI : 3000

در مجموع تعداد خطوط = 53,250
اگر حقوق هر برنامه نویس، ساعتی 20,000 باشد و به طور متوسط 200 ساعت در ماه کار کند، حقوق هر برنامه نویس در ماه 4,000,000 خواهد بود. همچنین اگر میانگین بهره وری LOC/p-h 1,000، هزینه هر خط کد برابر 4,000 است.
پس هزینه ی کل پروژه = تعداد خطوط * هزینه ی هر خط

Cost = 53250 * 4000 = 213,000,000

34. نحوه ی شناسایی ریسک ها

در اینجا بر اساس چک لیستی که در کتاب معرفی شده ریسک های سیستم خود را شناسایی میکنیم و در قسمت بعد آنها را بیان میکنیم.

آیا مدیران برنامه نرم افزاری و مشتری به صورت رسمی متعهد به حمایت از پروژه تند؟	بله
آیا کاربران نهایی مشتاقانه به این پروژه و محصول سیستم متعهد هستند؟	خیر
آیا نیازمندی ها به طور کامل توسط تیم مهندسی نرم افزار و مشتری ها شناسایی شده اند؟	بله
آیا مشتریان به طور کامل در تعریف نیازمندی ها درگیر شده اند؟	بله
آیا کاربران نهایی انتظارات معقولانه دارند؟	بله
آیا محدوده پروژه پایدار است؟	خیر
آیا تیم مهندسی نرم افزار ترکیبی از مهارت های مناسب را دارد؟	بله
تعداد افراد مشخص شده برای هر بخش کافی است ؟	بله
آیا شرایط پروژه پایدار است؟	خیر
آیا تیم پروژه با تکنولوژی که باید اجرا شود، تجربه دارد؟	خیر
آیا تعداد افراد در تیم پروژه کافی است تا این کار را انجام دهند؟	بله
آیا همه ذینفعان بر اهمیت پروژه و نیازمندی های سیستم / محصولی که ساخته می شود فق دارند؟	بله

35. بیان ریسک های پروژه و دسته بندی آن ها

طبق دسته بندی معرفی شده در کتاب پرسمن، انواع ریسک به صورت زیر است :

- Product size
- Business impact
- Stockholder characteristic

- Process definition
- Development environment
- Technology to be build
- Staff size and experience

همچنین طبق چک لیست ریسک معرفی شده در کتاب می توان ریسک ها را به صورت زیر دسته بندی کرد:

- Product size
- Business impact
- Process definition
- Staff size and experience
- Stockholder characteristic
- Development environment

در جدول زیر انواع ریسک های سیستم خود را با ذکر نوع آن مشاهده می کنیم.

category	risk
Product size	سیستم همه ی نیازمندی های کاربر را نداشته باشد.
Product size	نیازمندی های اضافی برای کاربر در نظر گرفته شده باشد.
Product size	نیازمندی های کاربر تغییر کند.
Stockholder characteristic	عدم جلب رضایت ذینفعان سیستم
Development environment	تغییر ابزار مورد استفاده در سیستم
Staff size	نیاز به نیروی کار بیشتر
Staff experience	نیاز به آموزش اعضای تیم (زبان جدید یا کار با ابزار جدید)
Business impact	بودجه ی پیش بینی شده کافی نباشد.
Product size	تخمین اشتباه بودجه (اضافی)

36. جدول ریسک های سیستم

اثرگذاری ریسک ها بین بازه 1 تا 4 به صورت زیر تعریف می شود :

1: بسیار تاثیر گذار

2: تاثیرگذار

3: محسوس

4: کم تاثیر

ریسک	دسته بندی	احتمال	impact
سیستم همه ی نیازمندی های کاربر را نداشته باشد.	Product size	30%	1
نیازمندی های اضافی برای کاربر در نظر گرفته شده باشد.	Product size	30%	2
نیازمندی های کاربر تغییر کند.	Product size	30%	2
عدم جلب رضایت ذینفعان سیستم	Stockholder characteristic	20%	1
تغییر ابزار مورد استفاده در سیستم	Development environment	40%	3
نیاز به نیروی کار بیشتر	Staff size	20%	1
نیاز به آموزش اعضای تیم (زبان جدید یا کار با ابزار جدید)	Staff experience	20%	3
بودجه ی پیش بینی شده کافی نباشد.	Business impact	50%	1
تخمین اشتباه بودجه (اضافی)	Product size	10%	4

37. برنامه مدیریت ریسک ها

ابتدا با استفاده از historical data هزینه ی هر ریسک را بدست می آوریم.
سپس اگر حاصل ضرب احتمال وقوع ریسک در هزینه بیشتر از یک ماکسیمم مقدار تعیین شده است یک risk information sheet برای مدیریت آن ریسک ایجاد می کنیم.
در اینجا ریسک های با هزینه ی بیش از 120000 را مدیریت می کنیم.

ریسک	هزینه	احتمال	هزینه*احتمال
سیستم همه ی نیازمندی های کاربر را نداشته باشد.	400000	30%	120000
نیازمندی های اضافی برای کاربر در نظر گرفته شده باشد.	300000	30%	90000
نیازمندی های کاربر تغییر کند.	300000	30%	90000
عدم جلب رضایت ذینفعان سیستم	500000	20%	100000

40000	40%	100000	تغییر ابزار مورد استفاده در سیستم
80000	20%	400000	نیاز به نیروی کار بیشتر
20000	20%	100000	نیاز به آموزش اعضای تیم (زبان جدید یا کار با ابزار جدید)
150000	50%	300000	بودجه ی پیش بینی شده کافی نباشد.
10000	10%	100000	تخمین اشتباه بودجه (اضافی)

risk information sheet :

Risk information sheet			
Risk ID : 1	Data : 1398/3/1	Prob : 30%	Impact : تاثیرگذار
Description : سیستم همه ی نیازمندی های کاربر را نداشته باشد.			
Refinement : Subcondition 1 : نیازمندی های کاریر به طور کامل شناسایی نشده باشند. Subcondition 2 : اشتباه در طراحی مدل های سیستم رخ داده باشد.			
Mitigation : 1. همکاری مداوم مشتری در روند پروژه بخصوص در شناسایی نیازمندی ها 2. در مرحله ی review حتما بررسی کنیم مدل طراحی تمام نیازمندی های مشتری را به درستی لحاظ کرده باشد.			
Management/ contingency plan/ trigger : ضمن در نظر گرفتن نیازمندی های جدید و ایجاد تغییر در سیستم در نظر بگیریم که همچنان امکان تغییر هست و سیستم تا جای ممکن تغییر پذیر کنیم. هزینه ی تغییر سیستم را به هزینه های سیستم اضافه کنیم .			
Current status : شروع mitigation			
Originator :مهدیس صفری		Assigned : -	

Risk information sheet			
Risk ID : 2	Data : 1398/3/1	Prob : 30%	Impact : بسیار تاثیر گذار
Description : بودجه ی پیش بینی شده کافی نباشد.			
Refinement : Subcondition 1 :			

نیازمندی های کاربر به طور کامل شناسایی نشده باشند. Subcondition 2 : حجم پروژه به درستی شناسایی نشده باشد
Mitigation : 1. نیازمندی ها به طور دقیق شناسایی شوند. 2. براساس milestone cost برنامه ی بودجه بندی را بررسی کنیم.
Management/ contingency plan/ trigger : در هر مرحله اگر هزینه سیستم از مقدار تخمین زده شده بیش از 5% انحراف داشت مجددا بودجه را تخمین بزنیم.
Current status : شروع mitigation
Originator :مهدیس صفری Assigned : -

Quality Checklist .38

Can content and/or function and/or navigation options be lored to the user's preferences?	No.
Can content and/or functionality be customized to the ndwidth at which the user communicates?	No.
Have graphics and other non-text media been used propriately? Are graphics file sizes optimized for display ficiency?	Yes.
Are tables organized and sized in a manner that makes them derstandable and displayed efficiently?	There is not any use of bles on this website.
Is HTML optimized to eliminate inefficiencies?	Yes,it is.
Is the overall page design easy to read and navigate?	Yes, it is.
Do all pointers provide links to information that is of interest to users?	No. No need on this website.
Is it likely that most links have persistence on the Web?	Yes, it is.
Is the WebApp instrumented with site management	No,it isn't.

utilities that include tools for usage tracking, link testing, local searching, and security?	

39. Build and buy

امروزه ابزارهایی وجود دارد که بسیاری از نیازمندی ها و ویژگی های سیستم را برآورده می کند.
این ابزارها تا حد زیادی اشکال زدایی(debug) شده است.
فروشنده می تواند آموزش، راهنمای کاربر و پشتیبانی مداوم ارائه دهد.
فروشنده به طور مرتب ابزار مربوطه را به روز رسانی کرده و بهبود می دهد.
اغلب یک جامعه کاربر ابزار مورد نظر وجود دارد که می تواند یک منبع برای حل مشکلات باشد.

از نظر زمانی معمولا build بیشتر طول می کشد؛ در صورتی که buy به شما اجازه می دهد تا سریعتر شروع به کار شوید.

نکته ی دیگری که اهمیت دارد این است که اعضای تیم برنامه نویسی چه قابلیت هایی دارند؟
بعنوان مثال تیم برنامه نویسی ای که تجربه فنی کمی دارند و هیچ مهارت پایگاه داده ای ندارند، در هر راه حلی نیاز به پشتیبانی خارجی(buy) دارد.

Build اجازه می دهد تا امکان طرحی مطابق با نیازمندی ها و اصلاح سیستم را داشته باشیم.
Buy ریسک های مانند مشکلات توسعه و رسیدن سر موعد ابزار مورد نظر توسط فروشنده دارد.

task	**buy**	**build**
Define requirements	yes	yes
Design		yes
Develop	configure	yes
Test / Debug	configuration only	yes

Perform regular aintenance		yes
Provide support		yes

در اینجا هدف سیستم انجام فرایندهایی است که آن را از دیگر سیستم ها متمایز می کند و همچنین در نظر گرفتن جدول بالا پس بهتر است Build را انتخاب کنیم.

چه پایگاه داده چه طراحی وب سایت همه توسط اعضای تیم برنامه نویسی، که مهارت های لازم را دارند، پیاده سازی می شوند این در صورتی است که برای نمونه نرم افزارهایی مانند wordpress برای طراحی وب سایت وجود دارند.