مقایسه XPو RUP

متدولوژی های توسعه ی نرم افزار به دو دسته تقسیم می شوند :

- متدولوژی های سنگین وزن (Heavy Weight)
- متدولوژی های سبک وزن (Light Weight) که شامل روشهای چابک (Agile) نیز می شوند.

این متدولوژی ها را می توان با توجه به معیارهای زیر مقایسه نمود :

✓ روش :

روش های سنگین وزن به صورت پیشگویانه (Predictive) عمل می کنند ولی روش های چابک سازگار(Adaptive) هستند.

✓ اندازه پروژه :

متدولوژی های سبک وزن بیشتر در پروژه های کوچک استفاده می شوند.

برای پروژه های بزرگ می بایستی از متدولوژی های سنگین وزن استفاده شود.

آمار نشان می دهد که تعداد پروژه های کوچک بسیار بیشتر است!

در متدولوژی های سنگین وزن مدیریت بصورت مطلق و استبدادی است در حالیکه مدیریت متدولوژی های سبک وزن بصورت غیر متمرکز و آزاد است .

✓ نحوه مستند سازی :

یکی از عیب های متدولوژی های سنگین وزن مستند سازیهای سنگین می باشد که کار بسیار دشوار و زمانبری می باشد و باید بصورت جامع و کامل انجام شود.
در حالیکه در متدولوژی های سبک وزن مستند سازی محدود وبسته به نیاز پروژه انجام می شود.

هر پروژه اي شرايط و مقتضيات خاص خود را دارد. لذا فرايند و متدولوژي مورد استفاده بايد جهت استفاده مناسب در آن پروژه خاص سفارشي سازي گردد؛ به اين عمل اصطلاحاً ***Tailoring*** مي گويند
بنابراين ميتوان اينطور گفت كه تشخيص نوع و دسته پروژه تأثير بسزايي در تشخيص نوع متدولوژي و همچنين سفارشي سازي آن متدولوژي و عناوين فرآورده ها و گزارشهاي خروجي پروژه دارد

دراین ارائه XP و RUP بر اساس ابعاد ذیل مورد مقایسه قرار گرفته اند :

- **Time and Effort Allocation**: در خصوص زمان و نفرات اختصاص داده شده به پروژه ها در هریک مقایسه انجام می گیرد .
- **Artifact:** تولیدات XPوRUPرا مقایسه می کند.
- **Activities**: در خصوص روشهائی که هر کدام از فرآیند ها،محصولات را تولید می کنند بحث می کند.
- **Disciplines:** روش هائی که در آن XP و RUP نواحی حائز اهمیت در مهندسی نرم افزار را معین می کنند مقایسه می کنند.
- **Roles:** تفاوت بین نقش ها (که Activityها را اجرا می کنند) در RUP ، و نقش هائی که نزدیک به موقعیت(Position) در یک تیم xp هستند را مشخص می کند.

در RUP جهت توسعه نرم افزار ، چرخه زندگی پروژه به فازهای Inception،Elaboration،Construction و Transition تقسیم شده است.هر فاز به چندین Iteration تقسیم می شودو هر Iteration ممکن است چندین builds نیاز داشته باشد

(ساخت (Build) يک فرايند مي‌باشد که در آن, هر يک از کامپوننت‌هاي سيستم، درون نرم‌افزار اصلي قرار گرفته تا بتواند مورد تست قرار گيرد. در واقع در اين فرايند اجزاي مختلف سيستم که توسط توسعه‌گران مختلف تيم, توليد شده است، به نرم‌افزار اصلي منتقل مي‌گردد)

در اکثر پروژه ها

در RUP:

- مدت زمان هر Iteration بین دو هفته تا ٦ ماه است
- تعداد Iteration ها در طول عمر پروژه بین ۳ تا ۹ عدد است
- بنابراین RUP پروژه هائی را پوشش می دهد که بین ۶ هفته تا ۵۴ ماه طول می کشد.

درXP :

- مدت ITERATION ها حدود دو هفته است
- مدت RELEASE ها ، ۲ ماه است (گاهی بیشتر)

از لحاظ طول دوره RELEASE هایXP مثل ITERATION های RUP هستند.(حداقل در طول ELABORATION وCONSTRUCTION ،برای پروژه هایی که مناسب XP هستند با تعداد نفرات تیم شامل ۱۰ نفر و بر اساس یک معماری مشخص)

در توسعه های بزرگ که RUP آنها را Handle می کند ولی XP نمی کند، Iteration های RUP بسیار طولانی هستند

در پروژه های بزرگ افراد در یک تیم دیده نمی شوندبلکه پروژه ترکیبی است از چندین تیم که هر یک روی زیر سیستم های متفاوت کار می کنند بنابراین می توان گفت هر یک از اینها شبیه سایز و scale پروژه های XP می باشد

در مثالهای کوچک iteration های RUP کم وبیش شبیه Releaseهای xp است وBuild های RUP کم وبیش شبیه iterationهای xp است.

فاز های RUP چگونه به XP،MAP می شوند؟

- ✓ در XP،BIG PLAN داریم که به RELEASE تقسیم می شود و آنها هم به ITERATION
- ✓ BIG PLAN کمک می کند که متوجه شویم انجام پروژه کار صحیحی است. بنابراین در XP قبل از شروع چرخه RELEASE وITERATION کاری انجام می شودکه امکانپذیری پروژه را بررسی می کند.که در RUP، فاز INCEPTION نامیده می شود.

- این بخش در XP خیلی سریع انجام می شود لیکن در RUP

تاجائیکه احتیاط اجازه دهدمی تواند طولانی شود(با توجه به ریسک)

فعالیت هایی که در XP دراین زمان انجام می پذیرد شامل :

- مقداری استخراج از نیازمندیها(نوشتن BIG STORIES)

- مقداری طراحی
- مقداری گفتگو با مشتری(انتظارات)
- تعامل شرایط BUSINESS،(FACTORING IN BUSINESS CONSTRUCT)

بنابراین دیده می شود که در طی چندین روز می توان به یک VISIONتوافق شده (که در XP ،BIG STORIES گفته می شود)و BUSINESS CASE رسید(بودجه و بازگشت سرمایه (ROIپروژه))

- پس inceptionتقریبا در هر دو برابر است. پس از آن xp مستقیم به طراحی اولین release می رود
- در RUP فاز elaboration بعد از inception می آید در iteration های elaboration، معماری راه حل پایدار می شود. در یک تفاوت آشکار ، Xpپیشنهاد می کند که طراحی release، " Function oriented" باشد بنابر این تمام Release ها یک ارزش business ای برای مشتری دارند.
- در Xp طراحی زیر ساخت تنها جهت پشتیبانی functionalityانتخاب شده جهت یک release می باشد و زیر ساخت بیشتربعدا هنگامیکه لازم باشد اضافه می شود و اگر سیستم به دلیل زیرساخت قبلی با شکست مواجه شود و تصمیم های قبلی با کارکرد های آتی معتبر نباشد،code دوباره جهت اینکه کار کند ساخته می شود(البته refactoring تنها در local improvement موثر است و اگر work flowکلی مشکل داشته باشد refactoring راه حل صحیح نمی باشد.)
- ایده آل این است که نباید زمان زیادی را صرف چیزی کرد که برای مشتری ارزشی ندارد.

در مقایسه چند نکته لازم است در RUP مورد نظر قرار گیرد:

- **مفهوم معماری در RUP حقیقتا زیر ساخت نمی باشد**:
- معماری یک سیستم نرم افزاری ساختاری از مولفه ای مهم سیستم است که از طریق Interface ها با هم در تعامل هستند و مولفه ها خود از مولفه های کوچکتر و Interface ها ساخته شده اند.بنابراین معماری نه تنها زیر ساخت بلکه همه راه حل را مورد توجه قرار می دهد.

نظریه معماری قابل اجرا در RUP باعث می شود برای مشتری ارزش BUSINESSای به همراه داشته باشد

- یک تصویر اولیه از راه حل فراهم می آورد که مشتری را درخصوص نیازمندیهای غیر FUNCTIONAL راضی می کند.(همچنین بعضی از نیازمندیهای Functional کلیدی در فرآیند)
- کاهش ریسک به این طریق با توجه به اینکه نیازمندیها NON FUNCTIONAL ریسکی هستند، ارزش BUSINESS ای قابل توجه ای برای CUSTOMER به همراه دارد.
- تاکید RUP بر روی معماری با این منظور است که مواردی را مورد توجه قرار دهد که در آن دستاورد اولیه ممکن است اشتباه باشد و نیاز است بطور کامل مجددا مورد بررسی قرار گیرد. برای سیستم های کوچکتر این موضوع ریسک کمتری دارد
- XP چیز کمی در مورد تغییرات مهم معماری میگوید و استفاده از آنها شجاعت می خواهد که یکی از ٤ ارزش در XP است.(شجاعت، اعتماد به انجام کار به صورت سریع و توسعهٔ مجدّد در صورت نیاز می‌باشد.)
- Refactoring جهت برخی از تغییرات بسیار سخت است . به عنوان مثال تبدیل یک سیستم SINGLE MACHINE و SINGLE USERبه یک سیستم چند کاره MULTI PROCESSOR که لازم است بسیاری از موارد REDESIGN شود

- هنگامیکه ریسک کمی ملاحظه شود و راه حل بتواند در چارچوب فعلی تحویل شود فاز elaboration در RUP (که با مباحث معماری سر و کار دارد)می تواند بسیار خلاصه باشد (و می تواند بیشتر خود را با استخراج و توصیف نیازمندیهای FUNCTIONAL درگیر کند). این است که چرخه حیات RUP به سمت چرخه حیات XP سقوط پیدا می کند.

- هنگامیکه مشکلی با موفقیت با با XP حل شود، توسعه RUPنیزمحصولی که تقریبا LIGHT WEIGHTشده می تواند داشته باشد (البته نه دقیقا مثل هم).

- فاز CONSTRUCTION در RUP معادل با سری های XP RELEASE است با این کیفیت که اگر شمااز RUP پیروی کنید باید تمام نتایج هر ITERATIONاز CONSTRUCTION را به مشتری تحویل دهید.

- XP فاز معادل با TRANSITION در RUP ندارد زیرا هر RELEASE XPدر دست مشتری است.

- در پایان پروژه نیز Death PROJECT داریم که فعالیت های PROJECT CLOSEOUT اتفاق می افتد.

چگونگی نگاشت فازهای XP بر RUP:

- فاز های XP (که هم در نسخه های XP(release) و چرخه های آن (Iteration)بکار برده می شوند) شامل Exploration(اکتشاف)، الزام،تعهد(commitment) و راهبری(steering) است. در سطح release، با مجموعه ای از فعالیت ها(که در RUP، WORK FLOW DETAILSنامیده می شوند) در دیسیپلین مدیریت پروژه در RUP در یک ردیف قرار می گیرند. آنها طرح هائی برای ITERATION بعدی(که با Exploration و commitment ،map می شوند) و پایش و کنترل پروژه (که با Steeing(راهبری) mapمی شوند) هستند.

فازهای XP:

- در خیلی از مراجع چرخه حیات پروژه XP را شامل فازهای ذیل می بیند:
- EXPLORATION(اکتساب)
- PLANNING طراحی
- ITERATION TO FIRST RELEASE
- PRODUCTIONIZING(فراوری)
- MAINTENANACE(نگهداری)
- DEATH
- نکته مهم اینست که هر RELEASE یک فاز اکتشاف دارد.بنابراین فاز اکتشاف که پروژه را به سمت اولین RELEASE هدایت می کند مورد خاصی است که در آن یک ورودی جهت عبور وجود دارد که درآن تصمیم گیری می شود که آیا ادامه کار عاقلانه است یا خیر . هر دوی XP RELEASEهاو ITERATION ها سه فاز دارند.
- EXPLORATION،COMMITMENTو STEERING. فاز نگهداری Maintenance در واقع بعد از اولین RELEASE معنی پیدا می کند.
- بطور نرمال در RUPاولین تحویل به مشتری در انتهای Construction انجام می پذیرد .البته این بدین معنا نمی باشد که این اولین باری است که مشتری محصول را دیده است ، در RUP مشتری جهت ارزیابی Iteration دعوت می شود و پروژه را می تواند تحت تست ببیند.

- در پروژه های سایز بزرگ آنقدر ریسک درمعماری راه حل وجود دارد که جهت تضمین پایدار کردن آن ۲ Iteration در elaboration قبل از سعی درساخت عملکرد نیازمندی های مشتری نجام می پذیرد. این بدین معنا ست که درانتهای elaboration تنها نیازمندی های در سطح معماری اکتشاف شده (و ممکن است پیاده سازی نیز شده باشد)

- XP این چنین عمل نمی کند . یکسری از راه حل های کامل در هر Release تحویل می دهد با فرض اینکه معماری بین Release ها با شکست مواجه نمی شود و یااگر چنین اتفاقی روی دهد Refactoring میتوان مشکل را رفع نماید . این اعتقاد وجود دارد که این موضوع کاربرد XP را محدود به کلاسی از سیستم ها می کند که بتوانند برروی یک معماری موجود از توانایی های شناخته شده ساخته شوند.

- هنگامیکه شرایط (تیم کوچک ، معماری مشخص ، توسعه ازنوع In-House ، استقرار با تشریفات کم (Low Ceremony Deployment)) وجود داشته باشد ، یک چرخه Tailor شده RUP می تواند همانند توسعه به روش XP عمل کند. البته Release اولیه ممکن است طولانی تراز حالت XP باشد لیکن ریسک کمتری وجود دارد .

- در مقابل هنگامی که شرایط (تیم های بزگ ، معماری جدید (بی سابقه)، توسعه هایی که با قرارداد های رسمی و اجبار به تحویل و استقرار) باشد می توان ادعا کرد که XP نمی تواند پاسخگو باشد.در حالیکه RUP از عهده چنین پروژه هایی به خوبی برمی آید.

- شاید دستآوردی شبیه XP (که از تکنیک های XP مانند Pair Programming و Refactoring استفاده می کند) بتواند دریک پروژه بزرگ استفاده شود لیکن زمانی امکانپذیر است که معماری پایدار باشد .

Artifact ها:

- RUP دارای بیش از ۱۰۰ ARTIFACT است که مورد این انتقاد قرار می گیردکه یک OVER HEAD غیر قابل تحمل به پروژه های کوچک تحمیل می کند لیکن این دید به دلایل زیر صحیح نمی باشد:

- در یک پروژه لازم نیست تمام ARTI FACTها تولید شود. انتخاب و TAILORING ، ARTI FACTها یک بخش ضروری از فرآیند است. RUP، راهنمائیهائی جهت اینکه کدام مستندات را می توان حذف کردو کدام یک را TAILOR کرد دارا می باشد.

- یک ARTI FACT(در RUP،توصیف ARTI FACT (Artifact Description)) یک موجودیت تشخیصی است.(SPECIFICATION) که اطلاعاتی که باید توسط یک فعالیت تولید شود را مشخص می کند

- می تواند به صورت مدل UML ، مستندات که شامل متن یا دیاگرام و یا حتی یک پیش نویس ساده روی WHITE BOARD نوشته شود. به هر حال به عنوان یک ARTI FACT در RUP به شمار می آید.

- به نظر می آید XP از انتقاد داشتن ARTI FACT های سنگین مصون باشد لیکن این موضوع به دلایل ذیل بیش از اندازه ساده گرفتن(OVER SIMPLIFACTION) است :

- XP به جهت اینکه مناسب نوع خاصی از توسعه باشد TAILORE شده است و با زیر مجموعه ای از دیسیپلین های RUP در مقیاس مطمئن در تعامل است بنابر این می توان انتظار داشت که درخواست ARTIFACT های کمتری باشد.

- XP بر روی STORY های کاربران و CODE تاکید دارد لیکن موارد دیگر که محصولات کار هستند هنگامیکه فرآیند توصیف می شود ذکر می گردد بنابراین تعداد ARTIFACTها به آن کمی که د ر نگاه اول به نظر می آید نمی باشد.

- در صورتیکه در RUP کلیه ARTIFACT را به صورت یک راهنمای UP- FRONT فراهم کرده است تا در زمان پروژه صرفه جوئی شود.

چرا XP کلیه ARTIFACT های RUP را لازم ندارد؟

- XP برنامه نویسی جهت برخوردارکردن نیازهای BUSINESS است. اینکه چگونه اتفاق می افتد،چگونه مدل می شود وچگونه CAPTUREمی شود نگرانی اصلی XP نمی باشد.
- تیم XP که CUSTOMER است رفع ابهامی از نیازمندیها را به صورت Story به تیم توسعه XP ارائه می دهد.
- آنها همچنین قضاوت کننده در خصوص ارزشهای BUSINESS هم هستند. اینکه این STORY ها چگونه به آن فرم بیان می شود مورد توجه XP نیست.
- به عنوان مثال آنچه که RUPدر دیسیپلین BUSINESS MODELING توصیف می کند خارج از SCOPE ، XP است.(BUSINESS MODEL در RUP دارای ١٤ARTIFACT است).
- منطق استقرار این RELEASEها مورد نگرانی توسعه (DEVELOPMENT) نمی باشد.
- بنابراین دیسیپلین های DEVELOPMENT در RUP به مقدار زیادی خارج از SCOPE ، XP است.
- (در RUP، DEPLOYMENT دارای ٩ ARTIFACT است) بنابر این برای پروژه های کوچک که با XP قابل پاسخگوئی است اگر بخواهیم RUP را TAILOR کنیم ٢٣ ARTIFACT حذف می شود.
- دلیل بعدی این است که Xp ادعا می کند گرفتن نیازمندیها و طراحی می تواند به ساد گی انجام پذیرد.
- نیازمندیها به صورت STORY های کاربر گرفته می شود گرفتن نیازمندیها و طراحی می تواند به سادگی انجام پذیرد.
- نیازمندیها به صورت STORY های کاربر گرفته می شود (که گاهی لازم است تقسیم شوند) سپس به TASK ها تجزیه می شوند، که در واقع طراحی است اما این برای همه سیستم ها عملی است وxp هم چنین ادعابی ندارد .
- رفتار سیستم های بزرگ و پیچیده بصورتی است که تفصیل آنها بدون دستاورد های سیستماتیک مثل USE CASE بسیار سخت است. همچنین توسعه ساختار ها در سیستم های پیچیده با کمک ABSTRACT VIEW از معماری سیستم امکانپذیر است.تعداد ١٤ عدد ARIFACTدر بخش نیازمندیها،١٧تا در RUP , ANALYSIS & DESIGNرا قادر می سازد تا از عهده پروژه هایی با سایز و پیچیدگی های متفاوت بر آید.
- در پروژه های کوچک در RUP تعداد ARTIFACT ها جهت (REQUIREMENT and ANALYSIS & DESIGN) به ٧ تا کاهش می یابد و تعداد ARTI FACT های مدیریت پروژه به ٦ تا کاهش یافته است.

مقایسه Artifact ها د رپروژه های کوچک

- بنابر این در خصوص پروژه های کوچک هنگامیکه ARTIFACT های RUP،TAILOR می باشد تعداد آنها به(٢٦-٣٠) تا کاهش می یابد."RUPآنچه را که XP نامفهوم و گنگ رها کرده است به خوبی مشخص می کند" و به شما اجازه می دهد تصمیم بگیرید چه چیزی ضروری است و راهنمائیها یی جهت انتخاب آنها ارائه می دهد.

Rough Mapping of XP artifacts to RUP artifacts

XP	RUP Small Project Roadmap
Stories Additional documentation from conversations	Vision Glossary Use-Case Model
Constraints	Supplementary Specifications
Acceptance tests Unit tests Test data Test results	Test Model
Software—code	Implementation Model
Releases	Product Release Notes
Metaphor	Software Architecture Document
Design—CRC, UML sketch Tasks Technical Tasks Design documents—produced at the end of the project Supporting documentation	Design Model
Coding standards	Design Guidelines Programming Guidelines
Workspace (development and other facilities) Testing framework tools	Tools
Release plan Story estimates Task estimates Iteration plan	Software Development Plan Iteration Plan
Overall plan—budget	Business Case Risk List Product Acceptance Plan
Reports on progress Task work time records Metrics data: Resources, Scope, Quality, Time Other metrics Tracking results Reports and notes on meetings	Status Assessment
Defects and associated data	Change Requests
Code management tools	Configuration Management Plan Project Repository Workspace
Spike	Prototypes
	Development Case Project-Specific Templates

:GUIDE LINES

- در ارتباط با ARTIFACT ها ، RUP راهنمائیهایی را ارائه می دهد که ذاتا اطلاعات بیشتری در خصوص ARTIFACT ها هستند(معنی آن،نحوه عرضه آن ،ارتباط با ARTIFACT های دیگر،محتوا و کاربرد آنها)

- در XP هم راهنمائیهایی در خصوص PRACTICE ها و ARTIFACT ها وجود دارد لیکن تلاشی جهت مدل کردن ارتباط آنها ندارد. علاوه بر این XP در Web Site های مختلفی پوشش داده شده است-جهت ثبت XP از نقطه نظرات مختلف و اضافه نمودن تجربه-این موضوع یک تفاوت دیگر بین XP و RUP را مشخص می کند.

- RUP یک محصو ل است اما XP نیست

- یک منبع معتبر برای XP وجود ندارد اگر چه کتابها شروع خوبی هستند لیکن نیازمند تکمیل کننده هایی می باشند.

فعالیت ها ACTIVITIES :

- RUP به طور رسمی ACTIVITYرا بعنوان کارهای انجام پذیرفته توسط یک نقش تعریف می کند.همچنین ARTIFACT های ورودی و ARTIFACT های جدید تولید شده و یا تغییر یافته را نیز مشخص می کند. سپس اینها مطابق با "دیسیپلین" و یا "فضائی که در گیر هستند"در پروژه ها دسته بندی می شوند این دیسسیپلین ها شامل موارد ذیل هستند:

- BUSINESS MODELING
- REQUIREMENTS
- ANALYSIS & DESIGN
- IMPLEMENTATION
- TEST
- DEPLOYMENT
- CONFIGURATIN& CHANGE MANAGEMENT
- PROJECT MANAGEMENT
- ENVIRONMENT

- زمان ACTIVITY بر اساس ARTIFACT هائی که تولید و مصرف می کنند می باشد.یک ACTIVITY می تواند بطور منطقی هنگامیکه ورودی های آن آماده است شروع شود. به این معنی که جفت های تولید کننده –مصرف کنندهACTIVITY ها می توانند در زمان OVERLAP داشته باشند اگر وضعیت ARTIFACT این اجازه را بدهد لازم نیست بطور متوالی باشند.
- هدف ACTIVITY ها در RUP اینست که از عدم شفافیت فرآیند درک تولید یک ARTIFACT بکاهند. با راهنمائیهای قدرتمندی که جهت چگونگی تولید هر چیز ارائه می دهند.
- همچنین ACTIVITY ها می توانند به مدیر پروژه در طراحی اش کمک کنند.
- RUP با استفاده از ACTIVITYها و ARTIFACT های مرتبط آنها،BEST PRACTICE ها را پشتیبانی می کند. توجه داشته باشید XPهم از عبارت PRACTICE استفاده می کند لیکن ارتباط دقیقی با مفهوم BEST PRACTICE وجود ندارد
- XP توسعه ساده نرم افزار را با استفاده از ۴ ACTIVITY اصلی زیر که بر اساس PRACTICE های آن هستند ارائه می دهد:

- CODING
- TESTING
- LISTENING
- DESIGNIN

- در واقع ACTIVITY های XP بیشتر نزدیک به دیسیپلین های RUP هستند تا ACTIVITYهای آن و بیشتر آنچه که برای یک پروژه XP اتفاق می افتد(علاوه بر کد نویسی،تست،گوش دادن،طراحی)از کاربرد PRACTICE های آن به دست می آید.

آیا ACTIVITY های یکسانی در XP و RUP وجود دارد؟

- بله وجود دارد لیکن ACTIVITYهای XP به صورت رسمی مشخص و تعریف نشده اند لیکن RUP بطور کامل و رسمی تعریف شده و ورودی ها و خروجی های آنها مشخص است.
- XP هم در روش تولید نقصی ندارد لیکن شاید به دلیل LIGHT WEIGHT ماندن،رسمیت و جزئیات حذف شده است. جزئیاتی که در RUP ارائه شده است می تواند هنگامیکه XP بر روی پروژه پیاده سازی می شود اضافه شوند.
- کمبود مشخص بودن(SPECIFICITY) قدرت و ضعف نمی باشد ولی نباید کمبود اطلاعات جزئی در XP را با سادگی اشتباه گرفت.
- از برخی نقطه نظرات،اعضاء پروژه نیاز دارند بدانند چه باید انجام دهند و درآن موقع نیازمند جزئیات هستند.

DECIPLINES AND WORKFLOW

- اخیرا RUP،عبارت DISCIPLINE را برای CORE WORK FLOW بکار می برد. یک دیسیپلین در RUP مجموعه ای از فعالیت ها است(و مفاهیم مرتبط) که مجموعه مشخصی از ARTIFACT ها را تولید می کنندکه جنبه های مهم توسعه نرم افزار را ارائه می دهند.
- دیسیپلین های RUP قبلا نام برده شده است آنها تمام جنبه هابی که یک سازمان یا BUSINESS هنگامیکه پرسنل را جهت توسعه،استقرار،کاربری،پشتیبانی،فروش و بازاریابی استخدام می کند و یا با نرم افزارهای بزرگ تعامل دارند نیاز دارد انجام دهد ، را پوشش نمی دهند.
- به عنوان مثال در حال حاضر مهندسی سیستم ها را پوشش نمی دهد و یا با تمام نیازمندیهای استاندارد های بین المللی فرآیند نرم افزار مثل ISO 15504 را پوشش نمی دهد
- XP صریحا بیش از این خود را محدود می کند.آن شامل ۴ ACTIVITY اصلی است که با استفاده از مجموعه ای از PRACTICE ها اجرا می شوند که نیازمند اجرای ACTIVITY های دیگری می باشند که به برخی از DISCIPLINE های RUP،MAP می شوند.

PRACTICE های XP شامل:

- **The Planning Game**: مشخص کردن محدوده RELEASE بعدی و یک طرح و برنامه برای آن به طور سریع با ترکیب اولویت های کاری و تخمین های فنی.
- **SMALL RELEASE:** یک سیستم را بطور سریع به وضعیت تولید ببرد و نسخه جدید را در یک چرخه کوتاه RELEASE می کند.
- **METAPHOR:** یک ویژن و زبان مشترک است که هم مشتري و هم تيم فني آن را درک مي‌کنند و براي تشريح پروژه استفاده مي‌شود
- **Simple DESIGN:** سیستم تا جائیکه ممکن است باید ساده طراحی شود. پیچیدگی اضافی به محض اینکه کشف شود حذف می گردد.
- **TESTING:** برنامه نویس ها به طور مستمر UNIT TEST هائی می نویسند که باید بدون عیب جهت ادامه توسعه اجرا شوند. مشتریان تست هائی می نویسند که نشان دهد FEATURE ها تمام شده اند.

- **REFACTORING:** برنامه نویس ها ساختار سیستم را بدون تغییر رفتارآن جهت حذف DUPLICATION،ارتقاء ارتباطات، سادگی و یا افزایش قابلیت انعطاف پذیری تغییر می دهند.
- **PAIR PROGRAMMING:** تمام کد های تولید توسط دو برنامه نویس پای یک سیستم نوشته می شود.
- **COLLECTIVE OWNERSHIP**: هر فردی هر کدی را در هر جائی از سیستم و هر زمانی می تواند تغییر دهد.
- **CONTINIOUS INTEGRATION:** سیستم را چندین بار در روز یکپارچه سازی می کنند و هر بار یک TASK کامل می شود.
- **40- HOUR WEEK:** کیفیت و کارايي مداوم عملكرد تيم، با اضافه‌كاري اعضاي تيم حفظ نمي‌شود. XP براي تضمين كيفيت، تعداد ساعت كاري منظم و مشخصي را تعيين مي‌كند.
- **ON-SITE CUSTOMER**: یک کاربر تمام وقت در تیم وجود دارد تا به سوالات پاسخ گوید.
- **CODING STANDARD:** تمام برنامه نویس ها کد ها را مطابق با قوانینی که به ارتباطات بین کد تاکید دارد می نویسند.
- ACTIVITY ها به عنوان نتیجه PRACTICE ها ایفای نقش می کنند. به عنوان مثال PLANNING Game با دیسیپلین PROJECT MANAGEMENT در MAP،RUP می شود.
- اگر چه برخی از عناوین آن خارج از XP،SCOPE است.به عنوان مثال BUSINESS MODELING استخراج نیازمندیها کاملا خارج از XP،SCOPE است .مشتری بطور ONLINE در کنار تیم نیازمندیها را تعریف می کند.(در فرم STORY).
- استقرار نرم افزار RELEASE شده خارج از XP،SCOPE است.
- به دلیل SCALEونوع استقرار که XP ارائه می دهد، می تواند با موضوعات دیسیپلین های محیط (ENVIRONMENT) و CONFIGURATION& CHANGE MANAGEMENT که RUP با جزئیات کامل پوشش می دهد ، خیلی سبک(LIGHTLY) تعامل داشته باشد.

استفاده از PRACTICE های XP در RUP:

- در دیسیپلین هائی که XP و RUP همپوشانی دارند،برخی از PRACTICE های XP می توانند در RUP بکار گرفته شوند(و برخی در حال حاضر هستند) به عنوان مثال:
-
- **PAIR PROGRAMMING**: در XPلازم است که محصول توسط بر نامه نویسانی که بطور دو تائی در پشت یک سیستم کار می کنند تولید شود و ادعا می کند که این کار بر روی کیفیت کد اثر داشته و هنگامیکه مهارت مورد نیاز است لذت بخش تر است.RUP تولید کد را در این سطح ریز توصیف نمی کندو بطور حتم این امکان وجود دارد که از PAIR PROGRAMMING در RUP استفاده شود. برخی اطلاعات مر بوط به این PRACTICE و TEST- FIRST DESIGN و REFACTORING در حال حاضر توسط RUP در فرم WHITE PAPERS فراهم شده است.
- به طور واضح این نیازمندی جهت استفاده این PRACTICE در RUP وجود ندارد و اعتقادی هم به لزوم آن نیست .
- مدارک موجود جهت اثبات مفید بودن در مقیاس صنایع بزرگ کم و اکثرا نقل قول است ، مطالعاتی در خصوص مقایسه برنامه نویسی دوتائی و تکی انجام پذیرفته است لیکن تیم های خوب اینگونه کار نمی کنند . در تیمی با فرهنگ ارتباطات باز هنگامی که نفرات در پرسیدن سوالات از همتاها احساس آزادی کنند و بر اساس فرآیند های تعریف شده کار کنند ، نمی توان خیلی فواید PAIR PROGRAMMING رامعین کرد.

- نفرات بدون اینکه مجبور باشند می توانند با هم کار کنند و مشکلاتشان را بطور طبیعی در یک کار تیمی حل کنند.
- البته در برخی شرائط کار کردن بصورت جفتی مزایائی دارد که می توانند به یکدیگر کمک کنند به عنوان مثال:
- در روز های اولیه شکل گیری تیم هنگامیکه نفرات می خواهند یکدیگر را بشناسند
- در تیم هائی که بر روی تکنولوژیهای جدید تجربه دارند
- در تیم هائی با ترکیبی از نفرات با تجربه و تازه کار
- **TEST FIRST DESIGN AND REFACTORING**
- اینها تکنیک های خوبی هستندکه می توانند در دیسیپلین پیاده سازی در RUP مورد استفاده قرار گیرند
- TEST FIRST DESIGNبه طور خاص روش بسیار خوبی جهت روشن نمودن نیازمندیها با جزئیات است
- خیلی از ACTIVITY های RUPمنفعت زیادی از داشتن مشتری در سایت به عنوان عضوی از تیم به دلیل افزایش اثر بخشی می برند . هنگامیکه مشتری به این شکل در دسترس باشد نیاز به تحویل اقلام میانی (بطور خاص مستندات) کاهش می یابد.
- هنگامیکه یک سیستم بخواهد تحویل داده شود حتی سیستم های کوچک به جز کد چیز های دیگری نیز لازم است تولید شود.
- XP در انتها اینکار را انجام می دهد اما بعنوان آنچه که بعدا تهیه می کند به عنوان مثال مستندات طراحی در آخر پروژه. در واقع تولید مستندات و ARTIFACT های دیگر رامنع نمی کند.(فقط در مورد آنها سکوت اختیار می کند) و می گویدتنها آنهائی را باید تولید کنید که به آن نیاز دارید. RUP هم موافق است ولی لیست آنها را می آورد و توصیف می کندچه چیزی را ممکن است لازم داشته باشید .
- Coding standard: RUP،ARTIFACT ای دارد (PROGRAMMING GUIDELINES) که تقریبا به عنوان اجباری هم تلقی می گردد.
- CONTINUOS INTEGRATION: RUP این مفهوم را از طریق BUILD در سطح زیر سیستم و سیستم پشتیبانی می کند(در یک ITERATION).

PRACTICE های XP که مقیاس پذیر نیستند((DONOT SCALE)

- برخی از PRACTICE ها مقیاس پذیر نیستند ودر RUP استفاده از آنها مشروط می شود.به عنوان مثال:
- COLLECTIVE OWNWRSHIP: داشتن یک تیم کوچک که اعضای آن مسئول یک سیستم کوچک و یا زیر سیستمی از سیستم بزرگ تر هستند و با تمام کد آن آشنا هستند خوب است.
- اینکه شما بخواهید تمام اعضای تیم بطور یکسان قدرت ایجاد تغییر در هر جا در کد را داشته باشند وابسته به طبیعت و پیچیدگی سیستم یا زیر سیستم است.معمولا اگر نفرات(ویا جفت ها)بر روی بخشی از کد که خودشان کار کرده اند تغییرات ایجاد کنند سریعتر و مطمئن تر است.
- آشنائی حتی با کد "نوشته شده به بهترین نحو "با گذشت زمان تقلیل می یابد مخصوصا اگر الگوریتم پیچیده ای داشته باشد
- REFACTORING: در سیستم های بزرگ REFACTORING بطور متناوب جایگزینی برای کمبود معماری نمی باشد. Beckمی گویداستراتژی طراحی xpمانند الگوریتم HILL-CLIMBING است. یک طراحی ساده انجام می دهید سپس کمی آن را پیچیده می کنید سپس کمی ساده تر وسپس کمی پیچیده تر. مشکل این الگوریتم بدست آوردن بهینه سازی LOCAL است .هنگامیکه تغییرات کم نمی توانند پیشرفتی ایجاد کنند ولی تغییرات زیاد می توانند.

- در RUP معماری دید رسیدن به "BIG HILL" را فراهم می کندو سیستم های بزرگ و پیچیده را قابل ردیابی می سازد.
- برای سیستم های بزرگ و پیچیده معماری بعنوان METAPHORE کافی نیست RUP چارچوب توصیفی بسیار غنی تر برای معماری فراهم میکند وآن تنها به عنوان معماری BOX های بزرگ و ارتباطات نیست.
- RAPID RELEASE: سرعتی که مشتری می تواند RELEASE جدید را پذیرفته و مستقر کند وابسته به خیلی از عوامل است: یکی از آنها سایز سیستم است که معمولا مرتبط با فشار BUSINESS است. یک چرخه ۲ ماهه برای برخی از کلاسهای سیستم ممکن است خیلی کم باشد منطق پیاده سازی مانع از آن می شود.
- در برخی از PRACTISEها که در نگاه اول به نظر می آید بالقوه قابل استفاده در RUP هستند نیازمند جزئیات بیشتر و احتیاط در هنگام کاربرد به طور عمومی است.
- SIMPLE DESIGN: XP بسیار FUNCTIONALITY محور است STORY های کاربر انتخاب می شودبه TASKها تجزیه می شود سپس پیاده سازی می شود. با توجه به اینکه BECK می گوید:

طراحی صحیح برای نرم افزار در هر زمانی آن است که:

- ۱-تمام تست ها اجرا شود.
- ۲-منطق تکراری نداشته باشد
- ۳-تمام اهداف برای برنامه نویس به صورت مهم شرح داده شود
- ۴-کمترین کلاس ها و متدهای ممکن را داشته باشد
- XP اعتقادی به اضافه کردن آنچه که در حال حاضر لازم نیست ندارد. مشکل در اینجا وجود چیزی شبیه مشکل بهینه سازی محلی در تعامل با کلاسی از نیازمندیها که در RUP،NON FUNCTIONAL نامیده می شود. این نیازمندیهای برای مشتری ارزش BUSINESS به همراه دارد ولی برای اینکه بخواهند از Story ها بیان شوند بسیار پیچیده هستند.
- RUP از طراحی آنچه بیش از نیاز است دفاع نمی کند اما از طراحی با یک مدل معماری در ذهن دفاع می کند که آن مدل معماری یکی از کلید های اصلی برای دسترسی به نیازمندیهای NON-FUNCTIONAL است.
- بنابراین RUP با Xp موافق است: طراحی ساده باید تمام تست ها را اجرا کند، با این الحاقیه که اینها شامل تست هائی که نشان می دهد نرم افزار نیازمندیهای NONFUNCTIONAL را هم دیده است. مجددا این موضوع هنگامیکه سایز و پیچیدگی سیستم افزایش می یابد، هنگامیکه معماری جدید است و یا هنگامیکه نیازمندیهای NON FUNCTIONALسنگین هستند مهم تر جلوه داده می شود.
- 40-HORSE WORD:همانند RUP،XP پیشنهاد می کند کار کردن بیش از وقت نبایدیک شرائط سخت باشد.XP محدودیت۴۰ساعت کارسخت را پیشنهاد نمی کندو Toleranceهای زمانهای کاری متفاوتی را مشخص می کند.
- مهندسان نرم افزار بدنام شده اند به اینکه ساعات طولانی بدون پاداش اضافه کار می کنند تنها برای راضی شدن به اینکه ببینند موضوعی کامل شده است و یک مدیر لزوما نیاز ندارد جهت متوقف ساختن آن مستبدانه عمل کند.
- آنچه که یک مدیر نباید انجام دهد استثمار کردن و یا تحمیل آن است و یک مدیر همواره باید متریک هایی بر روی ساعات واقعی کار جمع آوری کند اگر ساعات کاری هر فرد برای یک دوره زمانی زیاد به نظر می آید مطمئنا باید رسیدگی شود اما اینها موضوعاتی هستند که در شرائط خاص هنگامیکه رخ می دهد باید بین مدیر و نفرات حل شوند .۴۰ ساعت تنها یک راهنما است.(و البته محکم)

نقش ها:

در RUP ، فعالیت ها توسط ROLE ها اجرا می شوند . نقشها همچنین مسئول ARTIFACT های معینی هستند. نقش مسئول معمولا ARTIFACT را می سازد و مطمئن می شود که تغییرات ایجاد شده توسط نقشهای دیگر (اگر چنین تغییراتی اجازه داده شده باشد)باعث شکست ARTIFACT نمی شود. یک نقش در RUP می تواند توسط یک نفر و یا یک گروه انجام پذیرد. همچنین یک نفر یا گروه می توانند نقشهای متفاوتی را اجرا کنند.یک ROLE نباید به یک موقعیت در سازمان MAPشود ،نگاشت نقش ها در واحد های سازمانی باید MANY-TO-MANY باشد.

نقشهای RUP:

- RUP،۳۰ نقش تعریف می کند. نگاشت دقیقی بین نقش هاو دیسیپلین ها وجود ندارد.زیرا نقش به عنوان مثال معمار نرم افزار لزوما تنها به یک دیسیپلین محدود نشده است،اگر بخواهیم نقش ها را بر اساس آنچه که بطور عمده به آن تعلق دارند قرار دهیم:

- BUSINESS MODE- ۳ نقش
- REQUIREMENTS - ۵ نقش
- ANALYSIS &DESIGN- ۶ نقش
- IMPLEMENTATION- ۳ نقش
- TEST- ۲ نقش
- DEPLOYMENT- ۴ نقش
- CONFIGURATIN & CHANGE MANAGEMENT - ۲ نقش
- PROJECT MANAGEMENT- ۲ نقش
- ENVIRONMENT- ۳ نقش

چرا در RUP تعداد نقش های زیادی وجود دارد؟

- ROLE را در RUP جهت افراز فعالیت ها است و به طور ظریف و عالی تبعیض قائل شدن بین مهارتها و صلاحیت های مورد نیاز جهت اجرای نقش ها است که راهنمای انتخاب نفرات جهت اجرای نقش ها است. سطح تقسیم ،همچنین شناسائی موقعیت های جدید سازمانی را هنگامیکه اهمیت نقش تغییر می کند آسان می سازد.
- به عنوان مثال در یک پروژه کوچک غیر رسمی نقش مدیر پروژه و مدیر پیکر بندی (نقشهای RUP)میتواند توسط یک نفر انجام شود در یک پروژه بزرگ و رسمی کار مدیریت پیکربندی آنقدر می تواند خاص و سنگین باشد که نیاز به یک تیم کوچک جهت آن باشد . با توصیف ROLE ها به این روش ،RUP این نگاشت را ساده می کند.

- **نقشهای XP:**

- BECK ،۷ نقش را برای XP تعریف کرده است و سپس مسئولیت های آنها ،مهارتهاو ویژگیهای نفراتی که آنها را اجرا می کنند را مشخص کرده است. اینها شامل:

- PROGRAMMER-
- CUSTOMER-
- TESTER-

- TRACKER-
- COACH-
- CONSULTANT-
- BIG BOSS-
- علت اختلاف تعداد نقش هادر XP و RUP به سادگی بیان شده است:
  - XP تمام دیسیپلین های RUP را پوشش نمی دهد
  - -نقش هایXP به موقعیت ها نزدیکتر هستید تا به نقش های RUP به عنوان مثال برنامه نویس XP واقعا چندین نقش در RUP را اجرا می کند مثل CODE REVIEWER،INTEGRATION،IMPLEMENTATION که اینها صلاحیت های متفاوتی را نیز نیازمند هستند.
  - هنگامیکه نقش هایRUP در پروژه های کوچک MAP می شوندتعداد نقش های شبه XPکه موقعیت ها هستند کاهش می یابد.درROADMAP پروژه های کوچک تعداد موقعیت ها ۵ تا است همانند شکل زیر:

RUP Roles Mapped to a Small Project for ABC Company

ABC Company Job Title	RUP Role
Project Manager	Project Manager Process Engineer Deployment Manager Requirements Reviewer Architecture Reviewer
ABC Company Executive	Project Reviewer Stakeholder Requirements Reviewer
Chief Programmer	System Analyst Requirements Specifier User Interface Designer Software Architect Design Reviewer Process Engineer Tool Specialist Configuration Manager Change Control Manager *and, to a lesser extent, the same roles as the* Programmer
Programmer	Designer Implementer Code Reviewer Integrator Test Designer Tester
Administrative Assistant	*Responsible for:* • *maintaining the "Small Project" Web site* • *assisting the* Project Manager *role in planning or scheduling activities* • *assisting the* Change Control Manager *role in controlling changes to artifacts* • *may also provide assistance to other roles as necessary*

جمع بندی

مشکلات XP

- Business تنها توسط یک مشتری قابل ارائه نمی باشد
- کمبود توجه به نیازمندی های non functional
- کمبود لینک از Story به کد
- کمبود فرایند هایی جهت تولید تست ها عملکردی (Functional)
- کمبود فرایندهایی جهت تولید Story
- *XP زمانی کارایی دارد که کلیه شرایط ذیل مهیا باشد :*

- نیازی به Business Model نباشد و اعضای تیم مشتری نیازمندی ها را ارائه می کنند .
- نیازمندی های non Functional سنگین و سخت نمی باشد (آنقدر سخت هستند که نمی توان آنها را با Story بیان کرد)
- سایز تیم کمتر از ۲۰ نفر است
- تیم از نظر جغرافیایی پخش شده نمی باشد
- معماری شناخته شده است
- تنکنولوژی جدید نمی باشد
- شبیه توسعه های داخلی می باشد(به صورت فرمال قراردادی وجود ندارد، برای مشتری های خارج از مجموعه نمی باشد)

آنچه که خارج ار محدوده XP است :.

- Business Modeling و اهمیت abstraction
- استخراج نیازمندی ها
- استقرار
- معماری
- پروژه های بزرگ
- سیستم ها یی با تعداد زیاد Use Case
- Configuration management و Change management خیلی سبک است
- RUPیک محصول است ولی XP نیست
- کمبود اطلاعات در XP نباید با سادگی آن به اشتباه گرفته شود
- یک مشکل بزرگ RUP را تا کنون این مطلب می دانستند که حذف تهیه بخشهایی از RUP بوده که مورد استفاده پروژه نمی باشد لیکن این موضوع در نسخه های جدید تا حد بسیار بالایی برطرف شده است .
- مهمترین مزیت RUP این است که هیچ کدام از مطالب آن واقعا جدید نیست وبراساس best practice هایی است که در فیلدهای مختلف بارها استفاده و اثبات شده است . و این برخلاف XP است که اگرچه چیزی جدید نیست لیکن بر اساس Practice ها ی غیر پایدار است که با یکدیگرمورد استفاده قرار می گیرند تا از سرنگونی هر یک جلوگیری شود
- بنابراین طبق Best Practice هاي صنعت و تجربيات اخير حوزه نرم افزار، همه خبرگان و كارشناسان اتفاق نظر دارند كه در پروژه های کوچک باید از متدولوژیهای Agile و در پروژه های متوسط و بزرگ از متدولوژيهای مبتني بر فرايندهای يكپارچه بهره برد.
- متدولوژي‌هاي چابک به مستند سازي كمي نياز داشته، قابل تطبيق با تغييرات بوده و روي مشتري توجه و تمركز خاصي مي‌كنند

- *در متدولوژي XPتيم‌هاي كوچك يا متوسط (۱ تا ۱۰ نفر)، با تمركز و توجه خاص روي مشتري و با برنامه‌ريزي از قبل انجام شده در چرخه‌هاي كوتاه و سريع به كار توسعه مي‌پردازند و در كمترين زمان ممكن به مشتري خروجي تحويل مي‌دهند.*

- *اگر پروژه ای ابعادش کوچک باشد و ما به سمت استفاده از یک فرایند بسیار جامع، سنگین و پیچیده برویم، به جای اینکه فرایند و متدولوژی که هدف اصلي اش تضمین موفقیت پروژه است، به ما کمک نماید، مطمئناً باعث افزایش هزینه، افزایش منابع انساني مورد نیاز، تأخیر پروژه و گاهي نیز به شکست پروژه منجر خواهد شد.*

- *RUP متدولوژي بسياري منعطفي است و ميتوان با انجام Tailoring مناسب، از آن جهت استفاده در پروژه هاي حتي كوچك هم بهره برد. اما با توجه به اينكه RUP ماهيتاً براي پروژه هاي متوسط به بالا طراحي شده است و سفارشي سازي آن توسط پيمانكاران نيز، امكان دارد به خوبي انجام نشود و از طرفي متدولوژي هاي محبوب و مشهوری، نظیر متدولوژیهای Agile نیز در دنیا ظهور کرده اند، منطقي است كه در مورد پروژه های کوچک از این نوع متدولوژیها استفاده شود.*

- *از طرف ديگر هر چقدر پروژه بزرگتر و پيچيده تر شود، ما نياز داريم از قواعد و قوانين بيشتري براي اجراي كار بهره ببريم و اتكا به مكالمات شفاهي در كار كمتر ميشود.*

با افزايش ميزان حجم پروژه و پيچيدگيهاي آن، الزام و ضرورت استفاده از قوانين و دستورالعملهاي مدون، جهت اجراي پروژه بيشتر ميشود. بنابراين داشتن متدولوژي با فرايندهايي يكپارچه و منسجم كه از Best Practice هاي صنعت حاصل شده باشد، ضروری مي گردد.