

بسمه تعالی

درس مهندسی نرم افزار پیشرفته

فصل دوازدهم

روش های سریع الانتقال (چابک) توسعه نرم افزار

دکتر فریدون شمس

اهداف جلسه



- تقسیم‌بندی متدولوژی‌های توسعه نرم‌افزار
- معیارهای مقایسه متدولوژی‌ها با یکدیگر
- مقایسه متدولوژی‌ها بر اساس معیارهای مطرح شده
- اصول بنیادی متدولوژی‌های چابک
- معرفی روش‌های موجود

فهرست مطالب



- متدولوژی توسعه نرم افزار
- متدولوژی های سبک و سنگین
- مشکلات متدولوژی های سنگین
- مقایسه متدولوژی های سبک و سنگین
- تاریخچه متدولوژی های سبک وزن
- بیانیه چابکی
- معرفی چند متدولوژی چابک

متدولوژی توسعه نرم افزار



- فرآیند تولید و توسعه نرم افزار ذاتاً یک فرآیند بی نظم و پر هرج و مرج است. برای نظم دادن به این فرآیند، متدولوژی های توسعه نرم افزار مطرح شدند
- متدولوژی توسعه نرم افزار مشخص می کند، چه فرآورده ای (What) توسط چه کسی (Who) و در چه زمانی (When) تولید شود.

تقسیم بندی متدولوژی‌های توسعه نرم افزار



- متدولوژی‌های سنگین وزن (*Heavyweight*)

- فازها بطور کامل اجرا شده و مستندات کامل ایجاد می‌شود

- متدولوژی‌های سبک وزن (*Lightweight*)

- فازها به صورت کوتاه و مستندات به اندازه ایجاد می‌شوند

- متدولوژیهای چابک در دسته دوم قرار دارند

متدولوژی سنگین وزن



- در ۲۵ سال اخیر روش‌های بسیار زیادی برای توسعه نرم‌افزار معرفی شدند اما امروزه تعداد بسیار اندکی از آنها مورد استفاده قرار می‌گیرد
- متدولوژی‌های فعلی بیش از اندازه ماشین‌گرا و مکانیزه هستند و بصورت فرآیندی وارد جزئیات غیر ضروری می‌شوند، به همین دلیل این نوع متدولوژی‌ها را سنگین وزن می‌نامند

مشکلات متدولوژی‌های سنگین وزن



- مشتریان نرم‌افزارها حاضر نیستند که برای دست یافتن به نرم‌افزارهای مورد نیاز خود مدت زیادی منتظر بمانند
- رقابت بسیار شدید شرکت‌های تولید نرم‌افزار برای ارائه خدمات نرم‌افزاری به کاربران
- تغییرپذیری بسیار زیاد نرم‌افزارهای امروزی انکارناپذیر است

لزوم تغییرات در توسعه نرم افزار

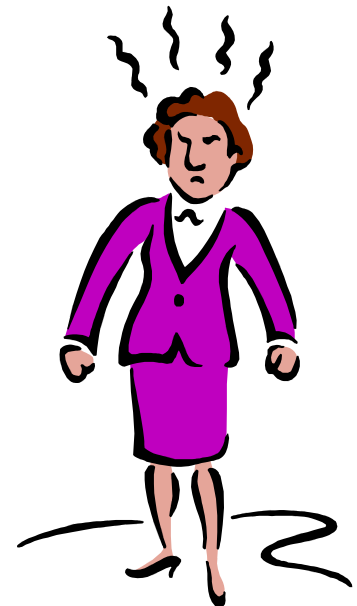
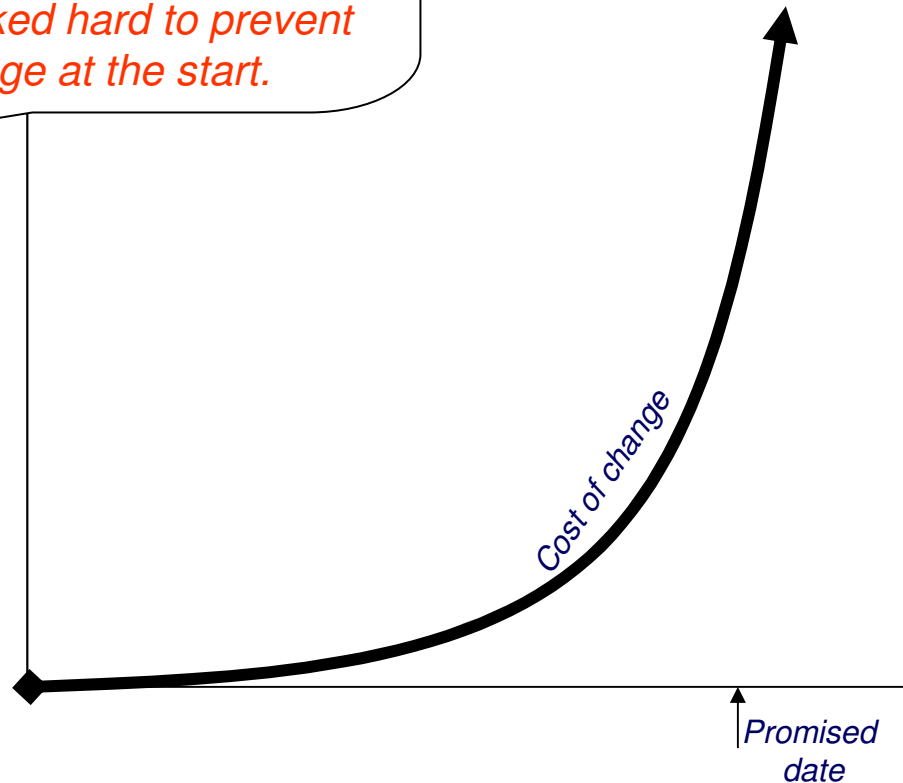


No Change!

- We are already running late.
- I need to meet my date.
- We worked hard to prevent change at the start.

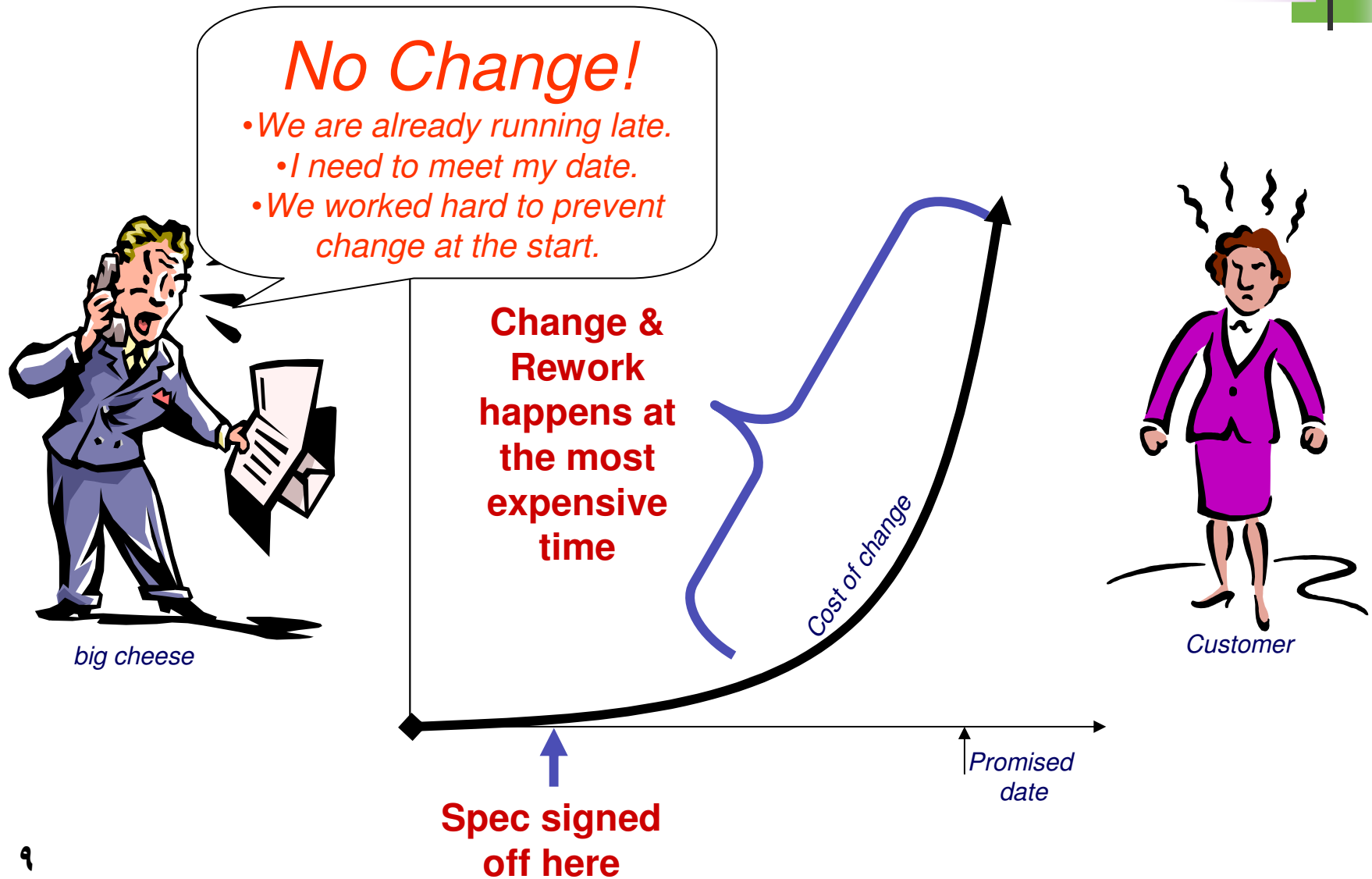


big cheese

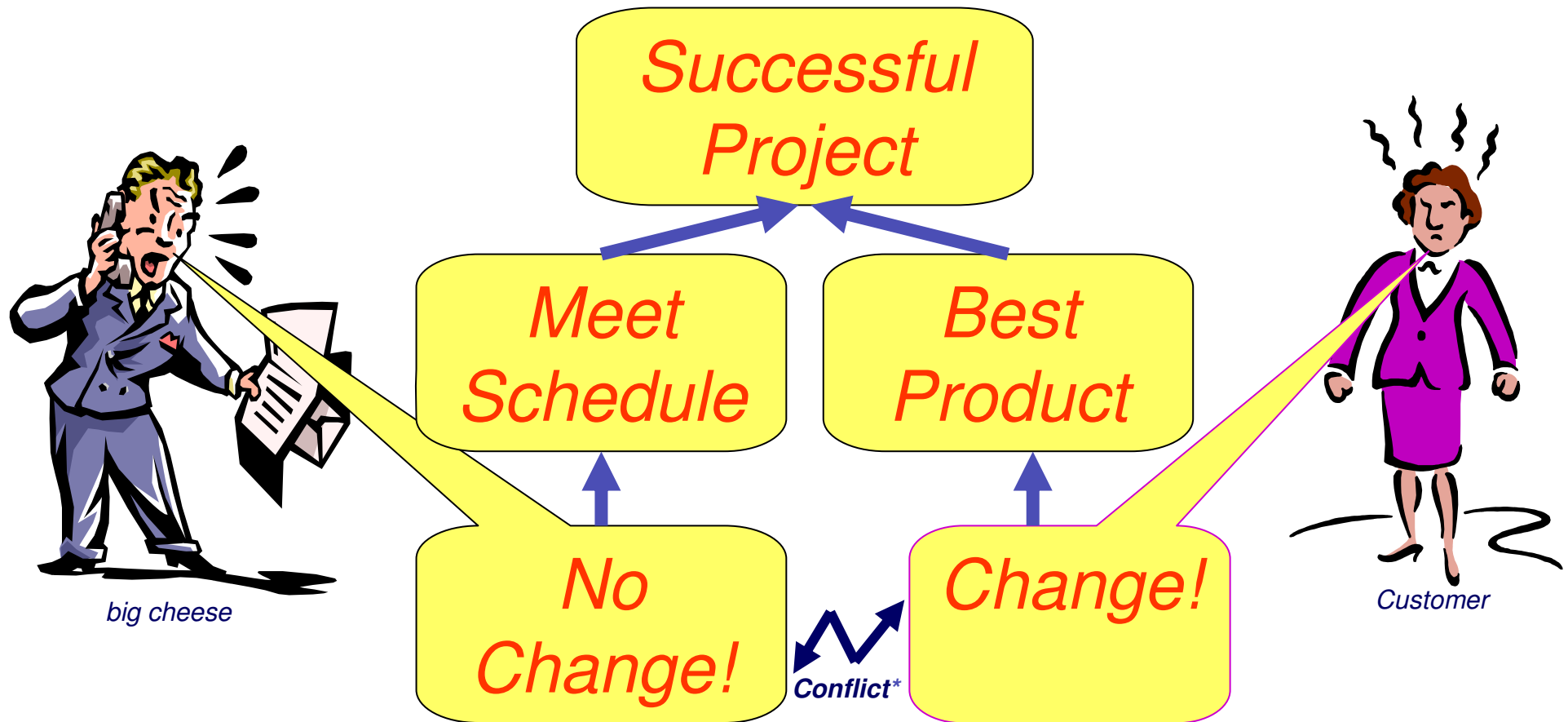


Customer

لزوم تغییرات در توسعه نرم افزار



لزوم تغییرات در توسعه نرم افزار



معیارهای مقایسه متدولوژی‌ها با یکدیگر



- روش
- معیار موفقیت
- اندازه پروژه
- سبک مدیریت
- نحوه مستندسازی
- چرخه‌ها
- اندازه تیم
- برگشت سرمایه

روش



■ روش‌های چابک بصورت *Adaptive* یا سازگار عمل می‌کنند
یعنی با شرایط منطبق می‌شوند

■ روش‌های سنگین وزن بصورت پیشگو یا *Predictive* عمل
می‌کنند یعنی در آغاز همه چیز را پیش‌بینی می‌کنند

Ad hoc,
Chaotic

Rigid,
Highly Structured

← ————— Process Discipline ————— →



◆ ————— CMM - SW ————— ◆



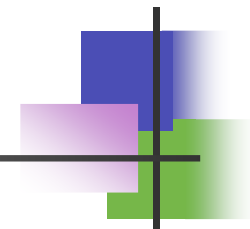
آیا همه چیز از ابتدا قابل پیش بینی است

معیار موفقیت



■ معیار موفقیت در روش‌های چابک دستیابی به ارزش کاری
(*Business Value*) است

■ در روش‌های سنگین وزن معیار موفقیت پیش رفتن در
راستای طرح اولیه است



روش‌های سنگین وزن انعطاف‌پذیری ندارند

اندازه پروژه



■ اندازه پروژه در روش‌های چابک کوچک است

■ اندازه پروژه در روش‌های سنگین وزن می‌تواند بسیار بزرگ باشد



این مسأله از محبوبیت روش‌های چابک نمی‌کاهد !!!
(آمار نشان می‌دهد که تعداد پروژه‌های کوچک بسیار بیشتر است)

سبک مدیریت



■ مدیریت در روش‌های چابک بصورت غیرمتمرکز و آزاد است

■ در روش‌های سنگین وزن مدیریت بصورت مطلق و استبدادی است

مدیریت غیرمتمرکز امکان تصمیم‌گیری بهتر را فراهم می‌کند

نحوه مستند سازی



- مستندسازی در روش‌های چابک بصورت بسیار محدود انجام می‌شود

- در روش‌های سنگین وزن مستندسازی بصورت کامل و جامع انجام می‌شود

در بسیاری از موارد مستند سازیهای سنگین، کار بسیار دشوار و زمانبری است

چرخه‌ها



- تعداد چرخه‌ها (*Cycles*) در روش‌های چابک بسیار زیاد است اما زمان آنها کوتاه است

- در روش‌های سنگین وزن تعداد چرخه‌ها کم است ولی زمان آنها بسیار زیاد است

زمانبر بودن چرخه‌های تولید ، موجب طولانی شدن زمان انتظار برای رسیدن به نشرها می شود

اندازه تیم



■ در روش‌های چابک اندازه تیم کوچک است (بین ۲۰ تا ۳۰ نفر)

■ در روش‌های سنگین وزن اندازه تیم توسعه بزرگ است

خلاقیت و همکاری در تیم کوچک بسیار بیشتر خواهد بود

برگشت سرمایه

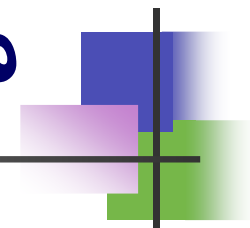


■ در روش‌های چابک سرمایه خیلی زود در طول پروژه بر می‌گردد

■ در روش‌های سنگین وزن برای برگشت سرمایه باید تا انتهای پروژه صبر کرد

روش‌های چابک از لحاظ اقتصادی بصرفه‌اند

مقایسه متدولوژی‌های سبک و سنگین



	Agile Methods	Heavy Methods
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project Size	Small	Large
Management Style	Decentralized	Autocratic
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Team Size	Small/Creative	Large

بیانیه روش‌های چابک



- در سال ۲۰۰۱ متخصصان روش‌های چابک بیانیه‌ای را منتشر کردند و این روش‌ها را در چهار اصل کلی به دنیای نرم‌افزار معرفی نمودند که عبارتند از:
 - **فردگرایی و تعامل** برتر از فرآیندها و ابزارها
 - **نرم‌افزار قابل اجرا** بهتر از مستندات مفهومی
 - **همکاری با مشتریان** بهتر از مذاکرات قراردادگرا
 - **پاسخ به تغییر** بهتر از دنباله‌روی از طرح

برخی روش‌های چابک



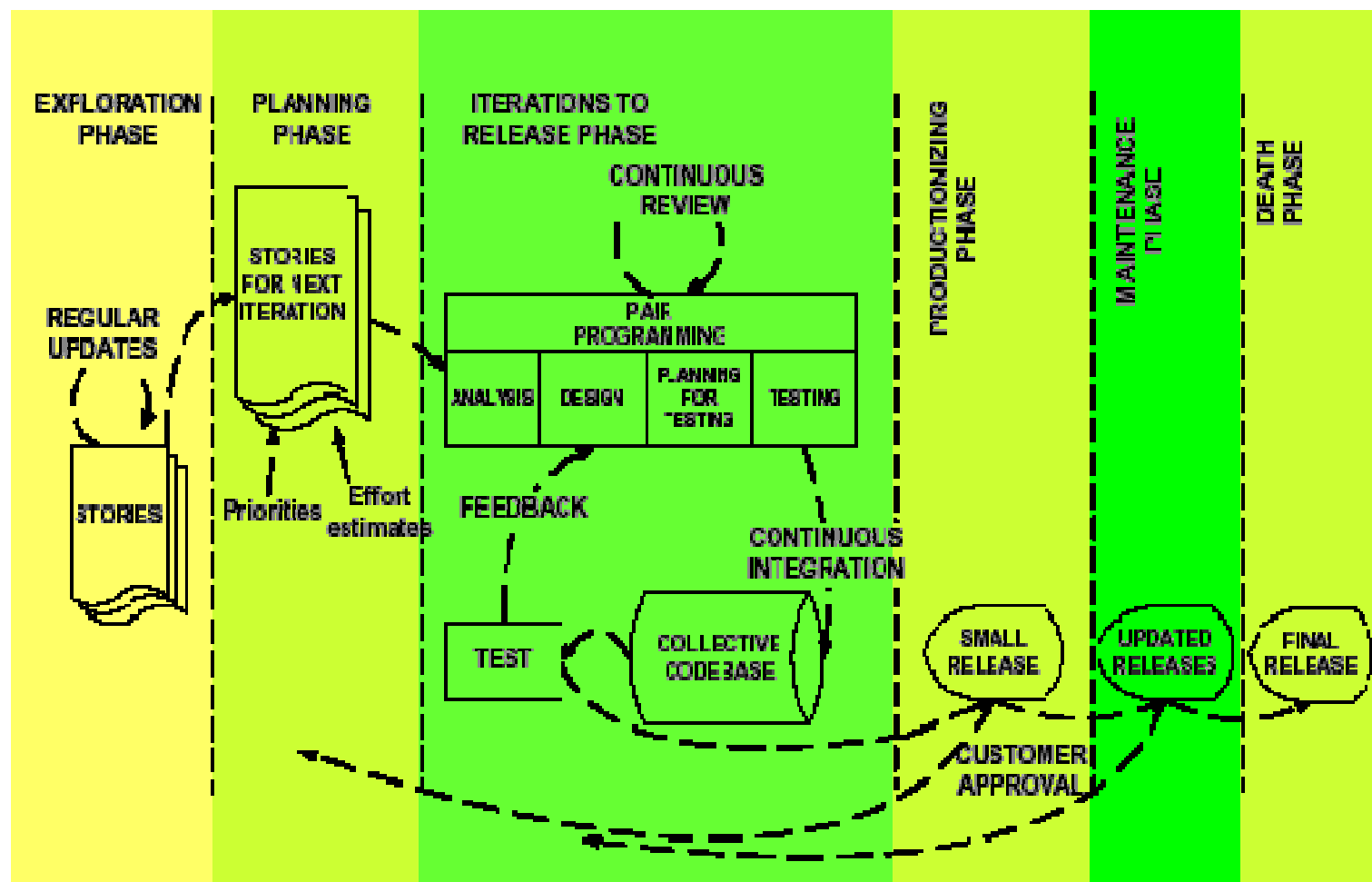
- **XP (*Extreme Programming*)**
- **Scrum**
- **Crystal Family**
- **FDD (*Feature Driven Development*)**
- **Dynamic System Development**
- **Adaptive Software Development**
- **Open Source Software Development**

متدولوژی XP (Extreme Programming)



- بر مبنای اصول سادگی، همکاری و بازخورد سریع استوار است
- ایده این روش متعلق به (۲۰۰۰) Kent Beck است
- این روش مبتنی بر آزمایش است (Test-Driven)
- نقش مشتریان در این روش بسیار پررنگ است
- فرآیند آن شامل ۱۲ فعالیت و ۵ فاز است

متدولوژی XP – چرخه حیات



متدولوژی XP – فازها



■ چرخه حیات XP شامل پنج فاز است

■ Exploration

■ Planning

■ Iterations To Release

■ Product Tionizing

■ Maintenance And Dead

متدولوژی XP – نقش‌ها و مسئولیت‌ها



- برنامه‌نویس
- مشتری
- آزمایش‌کننده
- پی‌گیری‌کننده (*Tracker*)
- مربی
- مشاور
- مدیر (رئیس ارشد)

متدولوژی XP – فرآورده‌ها



User Stories ■

- معمولاً بشکل متنی بوده و توسط مشتریان نوشته می‌شوند
- از طریق آنها نیازمندی‌های سیستم مشخص می‌شود

Iteration Plan ■

- مجموعه‌ای از User Story هاست که توسط مشتری انتخاب می‌شوند
- در یک تکرار که معمولاً دو هفته طول می‌کشد تولید می‌شود
- طرح‌های تکرار با توجه به اولویت مشخص شده توسط مشتری اجرا می‌شوند
- این انتخاب براساس بودجه تعیین شده توسط توسعه‌دهندگان خواهد بود

متدولوژی XP – فرآورده‌ها (ادامه)



Release Plan ■

- مجموعه ای از طرح‌های تکرار را در قالب یک نقشه کلی برای رسیدن به نشرها نمایش می‌دهد

Task ■

- زیرمجموعه‌ای از User Story ها هستند
- Task ها از نظر تکنیکی و کاری اولویت بیشتری دارند و باید سریع انجام شوند
- Task ها در مرحله طرح‌ریزی تکرار (Iteration Planning) مشخص می‌شوند

متدولوژی XP – فرآورده‌ها (ادامه)



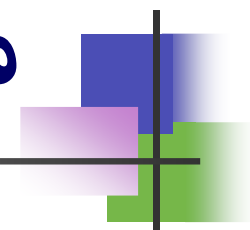
Metaphore

- نشان‌دهنده یک تصویر کلی از سیستم است
- برای هر عنصر در سیستم یک نام در نظر گرفته می‌شود
- ارتباط بین عناصر درگیر در سیستم از طریق Metaphore مشخص می‌شود

Spike

- یک راه‌حل ضربتی (Spike Solution)، برنامه ساده‌ایست که بوسیله آن می‌توان راه‌حل‌های بالقوه را کشف کرد
- در مواردی که User Story ها حساس و مهمند از آن استفاده می‌شود

متدولوژی XP – عملیات



Planning Game ■

- یک تعامل محصور (Close Interaction) بین مشتری و برنامه نویس بدست می آید
- برنامه نویس کار لازم برای پیاده سازی گزارش های مشتری را تخمین می زند و مشتری در مورد حوزه و زمان نشرها تصمیم گیری می کند

Simple Design ■

- تأکید اصلی در این روش بر روی طراحی ساده ترین راه حل ممکن است و پیچیدگی های غیرضروری و کدهای اضافی به سرعت حذف می شوند

متدولوژی XP – عملیات (ادامه)

Testing

- توسعه نرم افزار یک فرآیند آزمایش گراست (Test Driven). قبل از اینکه برنامه نویس یک خاصیت را اضافه کند، برای آن یک تست طراحی می کند که بصورت پیوسته اجرا می گردد

Refactoring

- بازسازی سیستم با حذف موارد تکراری، بهبود ارتباطات، ساده سازی و افزایش انعطاف پذیری سیستم

Pair Programming

- دو نفر کد را روی یک کامپیوتر می نویسند (یک کدنویس و یک متخصص استراتژی)

متدولوژی XP – عملیات (ادامه)



Collective Ownership ■

- هر فردی می تواند کد را در هر زمانی تغییر دهد


Continuous Integration ■

- یک تکه کد جدید در حداقل زمان ممکن به کد اولیه می پیوندد، بنابراین سیستم دفعات زیادی در روز یکپارچه شده و ساخته می شود

40 Hour Week ■

- حداکثر چهل ساعت کار در هفته کافی است. این مورد اجباری است و بیشتر از این ساعات کار مجاز نمی باشد.

متدولوژی XP – عملیات (ادامه)



On- Site Customer ■

■ مشتری باید بصورت تمام وقت برای تیم توسعه در دسترس باشد

Coding Standards ■

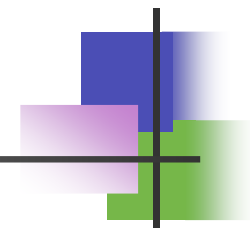
■ قواعد کدنویسی باید توسط برنامه‌نویسان رعایت شود و ارتباط بین کدها مورد توجه قرار گیرد

متدولوژی FDD (Feature Driven Development)

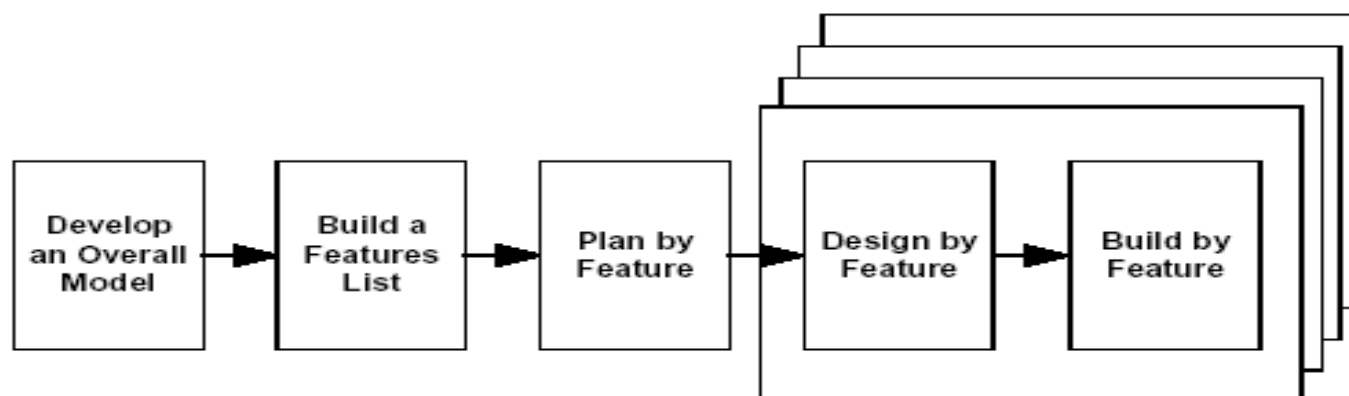


- تمام فرآیند توسعه نرم افزار را پوشش نمی دهد و بیشتر روی دو فاز **طراحی و پیاده سازی** متمرکز می شود
- برای استفاده به همراه سایر فعالیت های یک پروژه توسعه نرم افزار طراحی شده است و هیچ مدل فرآیند خاصی لازم ندارد
- مبتنی بر توسعه **تکراری** با انتخاب بهترین و موثرترین فعالیت ها است
- روی جنبه های کیفیتی تأکید دارد و شامل **نشرهای محسوس** و پیگیری دقیق پیشرفت پروژه است

فرآیندهای FDD



- شامل پنج فرآیند ترتیبی است که از طریق آنها فعالیت‌های طراحی و پیاده‌سازی انجام می‌شود



- قسمت تکراری فرآیند FDD (طراحی و ساخت) از توسعه چابک حمایت می‌کند
- هر تکرار از یک خاصیت، معمولاً ۲ تا ۳ هفته زمان می‌برد

متدولوژی FDD – نقش‌ها




■ FDD نقش‌های خود را به سه دسته کلی تقسیم می‌کند

■ نقش‌های کلیدی

■ نقش‌های حمایتی


■ نقش‌های اضافی

متدولوژی FDD – نقش‌های کلیدی



- مدیر پروژه
- معمار اصلی
- مدیر توسعه
- برنامه نویس (*Deployer*)
- متخصص دامنه (*Domain Manager*)

متدولوژی FDD – نقش‌های حمایتی



- مدیر نشر
- مشاور زبان (*Language Guru*)
- مهندس ساخت (*Build Engineer*)
- مسئول ابزار (*Toolsmith*)
- مدیر سیستم

متدولوژی FDD – نقش‌های اضافی



■ سه نقش اضافی که در همه پروژه‌ها وجود دارند

■ آزمایش کننده

■ مستقر کننده

■ نویسنده فنی (*Technical Writer*)

■ هر عضو می‌تواند چندین نقش بازی کند و هر نقش ممکن

است به چند عضو نسبت داده شود

متدولوژی FDD – فعالیت‌ها



Domain Object Modeling

- شامل استخراج و توضیح دامنه مسأله می‌باشد

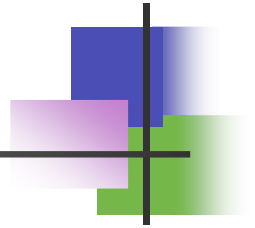
Developing By Feature

- توسعه و بررسی میزان پیشرفت پروژه از طریق دنبال کردن پیاده‌سازی لیست وظایف و خواص مشخص شده

Individual Class Ownership

- برای هر کلاس یک شخص وجود دارد که مسئول سازگاری، کارایی و صحت آن است

متدولوژی Scrum



- از یک استراتژی در نوعی توپ بازی گرفته شده است
- تأکید روی اصول **انعطاف پذیری**، **سازگاری** و **سودمندی** است
- **تمرکز**: چگونه اعضای تیم باید عمل کنند تا سیستم تولید شده، در یک محیط کاملاً تغییر پذیر، انعطاف پذیری کافی داشته باشد
- **ایده اصلی**: توسعه سیستمها شامل چندین متغیر محیطی و تکنیکی است (نیازها، زمان، منابع و تکنولوژی) که احتمالاً در طول فرآیند توسعه تغییر می کنند. این مطلب فرآیند توسعه را پیچیده و غیر قابل پیش بینی می کند

متدولوژی Scrum – فرآورده‌ها




■ فرآورده‌های *Scrum* به سه دسته اصلی تقسیم می‌شوند

Product Backlog ■

Sprint Backlog ■

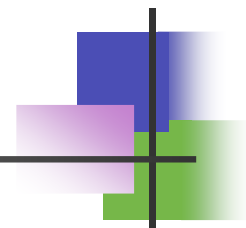
Sprint BurnDown Chart ■

متدولوژی Product Backlog – Scrum



- شامل یک **صف اولویت‌بندی** شده است که در آن وظیفه‌مندیهای تکنیکی و کاری نمایش داده شده‌اند که باید توسعه داده شوند
- برای هر مورد مشخص شده در این فرآورده، خواصی مانند وضعیت، اولویت و تخمین کاری وجود دارد

متدولوژی Sprint Backlog – Scrum

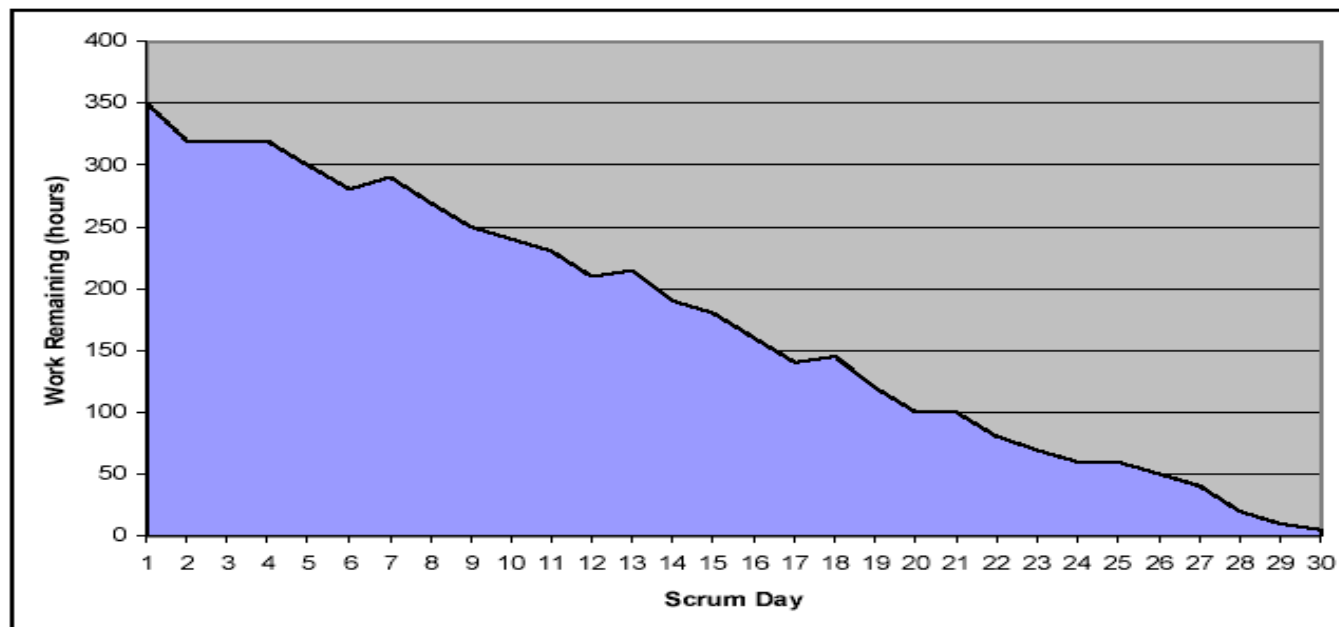


- مجموعه‌ای از **موارد حرفه و فنی** که برای تکرار جاری (Current Iteration) زمانبندی شده‌اند در این فرآورده نمایش داده می‌شوند
- **نیازها** در این فرآورده به **وظایف** تبدیل می‌شوند
- برای هر وظیفه یک **شرح کوتاه** وجود دارد و مشخص می‌شود که چه کسی مسئول انجام آن است و همچنین وضعیت و تعداد ساعات باقیمانده تا تکمیل شدن هر وظیفه در این فرآورده مشخص می‌شود

متدولوژی Scrum – Sprint BurnDown Chart



- ساعات باقیمانده برای تکمیل شدن همه وظایف مربوط به یک Sprint را در قالب یک گراف بصورت برجسته نمایش می دهد
- در شکل زیر یک مثال ساده از آن دیده می شود

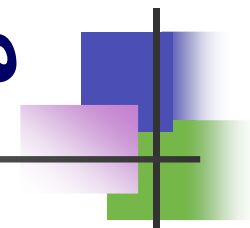


متدولوژی Scrum – نقش‌ها



- Scrum Master
- Product owner
- Scrum team
- Manager

متدولوژی‌های خانواده Crystal



- شامل مجموعه‌ای از متدولوژی‌های متفاوت است که مناسبترین آنها برای هر پروژه منحصر به فرد انتخاب می‌شود
- دارای اصولی است که متدولوژی‌ها را برای شرایط مختلف موجود در پروژه‌ها سفارشی می‌کند
- روش Crystal پیشنهاد می‌کند که یک متدولوژی مناسب براساس اندازه و میزان بحرانی بودن پروژه انتخاب شود

متدولوژی های خانواده Crystal (ادامه)



هر عضو از خانواده Crystal با یک رنگ مشخص می شود که میزان سنگینی متدولوژی را نشان می دهد. رنگ تاریکتر نشان دهنده متدولوژی سنگین تر است

Criticality of the system	L6	L20	L40	L80
	E6	E20	E40	E80
	D6	D20	D40	D80
	C6	C20	C40	C80
	Clear	Yellow	Orange	Red
	Size of the project			

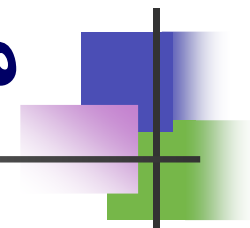
C: Comfort

D: Discretionary Money

E: Essential Money

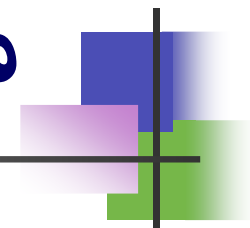
L: Life

متدولوژی‌های خانواده Crystal (ادامه)



- تمامی پروژه‌ها از توسعه **افزایشی** با بازه زمانی حداکثر ۴ ماه استفاده می‌کنند
- تأکید روی **ارتباطات و همکاری** بین افراد درگیر در پروژه است
- هیچ فعالیت یا ابزاری را برای توسعه محدود نمی‌کند. مثلاً می‌توان از فعالیت‌های XP و Scrum با هم در این روش استفاده کرد

مزایای روش‌های سریع الانتقال



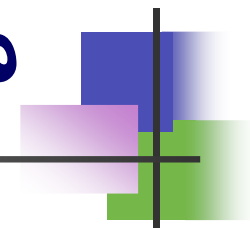
- تأکید روی شرکت‌دادن مشتری در پروژه‌ها است که در پروژه‌های کاربردی بسیار مفید است
- تأکید روی کارگروهی و ارتباط متقابل که در بالا بردن راندمان کاری نقش مهمی دارد
- همه افراد درگیر در پروژه در قبال کیفیت محصول مسئولند
- سنجش مستمر کارهای انجام شده از مزایای بسیار مفید این روش‌ها است

مزایای روش‌های سریع الانتقال (ادامه)



- توسعه افزایشی که با روش‌های مدرن توسعه نرم‌افزار سازگار است
- طراحی ساده و روشن برای فرآیند توسعه
- بازبینی‌های مستمر که به بالا رفتن کیفیت کار برنامه‌نویسان کمک می‌کند

معایب روش‌های سریع الانتقال



- **بدلیل کمبود فعالیت‌های طراحی** در این روش‌ها، اگر کد بیش از چند هزار خط باشد ممکن است فرآیند توسعه با موانع خطرناکی برخورد کند
- **کمبود مستندات مربوط به طراحی** در این روش‌ها آنها را به پروژه‌های کوچک محدود می‌کند و قابلیت‌های استفاده مجدد را در آنها محدود می‌کند
- **کمبود فرآیندهای بازبینی ساخت یافته**

معایب روش‌های سریع الانتقال (ادامه)



- کمبود فرآیند طراحی منظم و استفاده از بازبینی‌های غیر ساخت یافته باعث اتلاف زمان و هزینه می‌شود
- کمبود طرح کیفیت. در این روش‌ها هیچ نوع طرح استاندارد برای ارزیابی کیفیت وجود ندارد
- کمبود راهنماهای آموزشی برای استفاده از این روش‌ها