فصل سوم

- **Agile Development**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e

by Roger S. Pressman

Slides copyright © 1996, 2001, 2005, 2009 by Roger S. Pressman

For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 7/e.* Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

ایده اصلی توسعه چالاک

- ”ما راه‌های بهتری را برای توسعه نرم‌افزار با انجام آن و کمک به دیگران برای انجام آن، نشان خواهیم داد“
- در طی این کار ما به ارزش‌های زیر دست یافته ایم :
  - اهمیت اشخاص و ارتباطات نسبت به فرایند و ابزار
  - اهمیت نرم‌افزار عملیاتی نسبت به مستندات کامل
  - اهمیت همکاری مشتری نسبت به مذاکرات در حین قرارداد
  - اهمیت پاسخ به تغییرات نسبت به پیگیری یک طرح اولیه
- با توجه به مطالب بالا، درحالی که مسائل مطرح شده در سمت چپ جملات دارای اهمیت می‌باشند، مسائل مطرح شده در سمت راست از اهمیت بالاتری برخوردارند.

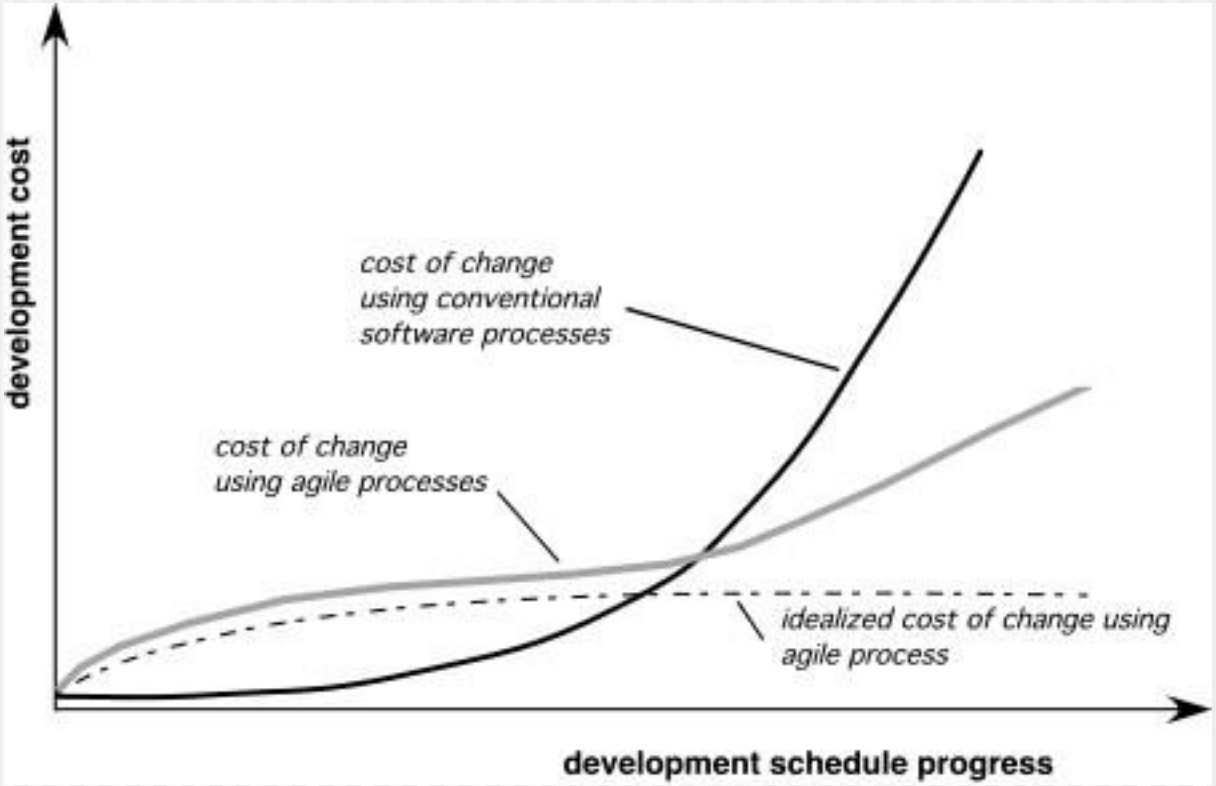
چالاکی به چه معنی می باشد؟

- پاسخگویی موثر (سریع و سازگار) به تغییرات
- ارتباطات موثر میان همه ذینفعان
- قراردادن مشتری در تیم نرم‌افزاری
- تشکیل یک تیم برای اینکه کار انجام شده تحت کنترل باشد

و این موارد موجب می‌شود :

- تحویل سریع و افزایشی (Incremental) نرم‌افزار

چالاکی و هزینه تغییرات



فرایند چالاک

- از نیازهای ارائه شده (سناریوها) توسط مشتری نشات می‌گیرد.
- طرح‌ها، کوتاه مدت هستند.
- توسعه نرم‌افزار به صورت چرخشی با تاکید زیاد بر روی فعالیت‌های ساخت سیستم
- تحویل نرم افزار در چندین افزایش (Increment)
- در صورت تغییر، با تغییرات تطابق می‌یابد.

اصول چالاکی

۱. بالاترین اولویت، رضایت مشتری از طریق تحویل به موقع و مداوم نرم افزار ارزشمند می باشد.

۲. تغییرات در نیازها را بپذیرید، حتی در اواخر توسعه. فرآیندهای چالاک تغییر را در جهت بهره مندی مشتری مهار میکنند.

۳. ارائه نرم افزار قابل اجرا بصورت مکرر،از چند هفته تا چند ماه، با اولویت زمانبندی کوتاه تر است.

۴. تعامل کاربران و توسعه دهندگان بصورت روزانه در طول پروژه.

۵. پروژه را بر پایه افراد با انگیزه شکل دهید، نیازها و محیط مناسب را برای آن‌ها فراهم کنید، به آن‌ها اعتماد کرده و اجازه دهید کار را انجام دهند.

۶. بهترین راه برای بیان اطلاعات در تیم نرم‌افزار مباحثه رو در رو می‌باشد.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

اصول چالاکی

۷. نرم‌افزار عملیاتی اولین معیار اندازه‌گیری پیشرفت است.

۸. توسعه نرم‌افزار با همکاری مسئولین، توسعه دهندگان، و کاربران انجام می‌گیرد.

۹. توجه مستمر به تعالی فنی و طراحی خوب موجب افزایش چالاکی می‌شود.

۱۰. سادگی یک اصل اولیه می‌باشد.

۱۱. بهترین معماری، نیازها و طراحی از تیم‌های Self-Organizing به دست می‌آید.

۱۲. در بازه‌های زمانی مشخصی تیم‌ها بررسی می‌کنند که چگونه کارایی خود را افزایش دهند.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

عوامل انسانی

- *فرآیند به نیازهای افراد و تیم وابسته است و نه چیز دیگر*
- *چند پارامتر کلیدی باید بین افراد تیم چالاک و خود تیم وجود داشته باشد*
  - **رقابت**
  - **تمرکز مشترک**
  - **همکاری**
  - **توانایی تصمیم گیری**
  - **توانایی حل مسئله فازی**
  - **احترام و اعتماد متقابل**
  - **خودسازماندهی**

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

برنامه نویسی افراطی(XP)

- فرآیند چالاک بسیار گسترده که اولین بار توسط Kent Beck پیشنهاد شد .
- برنامه ریزی
  - ارائه User Stories
  - ارزیابی و اندازه گیری داستان هاو اختصاص هزینه به هریک
  - گروه بندی داستانها برای تحویل افزایشی
  - تعیین تاریخ تحویل
  - بعد از اولین تحویل و مشخص شدن سرعت پروژه، تاریخ‌های تحویل افزایش‌های بعدی نیز مشخص می‌شود.

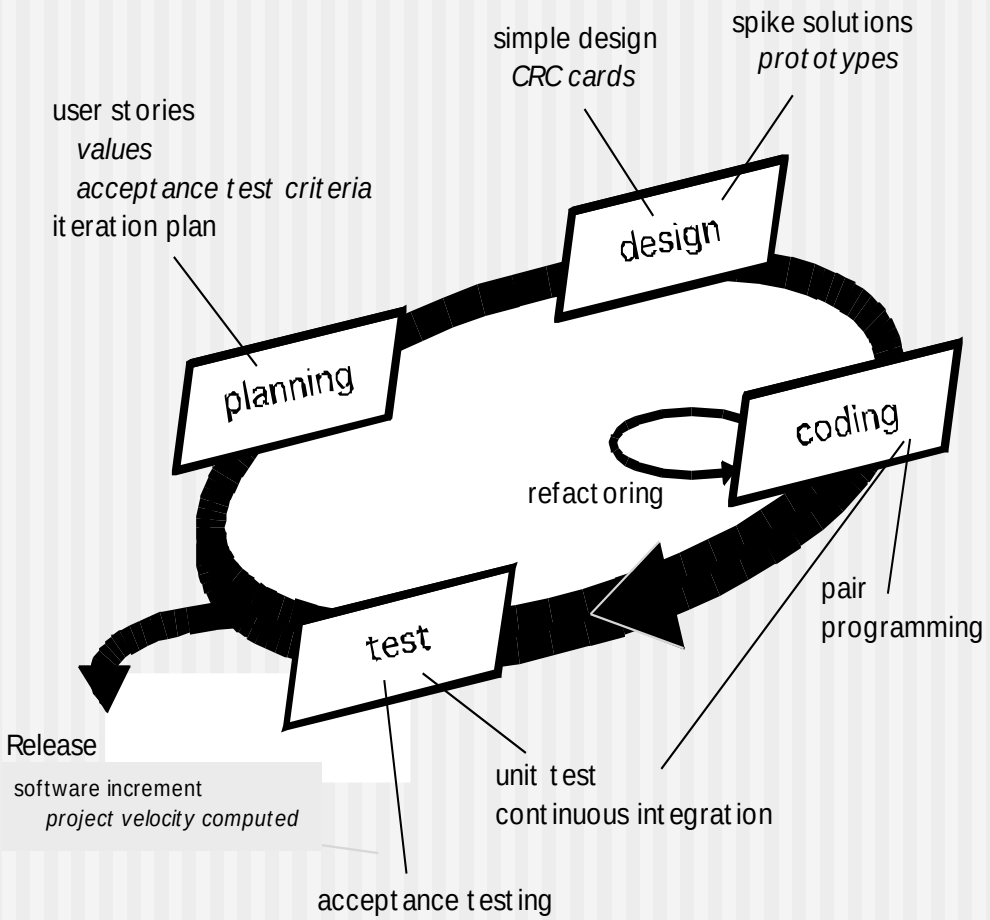
These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

برنامه نویسی افراطی(XP)

- طراحی در XP
  - بر مبنای قانون KIS (Keep it Simple)
  - توصیه به استفاده از کارت‌های CRC
  - برای مسائل طراحی پیچیده، نمونه ایجاد شود.
  - توصیه به انجام Refactoring (بهبود افزایشی طراحی برنامه)
- کد نویسی در XP
  - توصیه به تولید آزمایش های واحد قبل از آغاز کد نویسی
  - توصیه به جفت برنامه نویسی
- تست در XP
  - انجام آزمایش واحد به صورت روزانه
  - آزمایش پذیرش توسط مشتری تعریف شده و با آزمایش موارد قابل مشاهده توسط وی، انجام می‌گیرد.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

برنامه نویسی افراطی(XP)



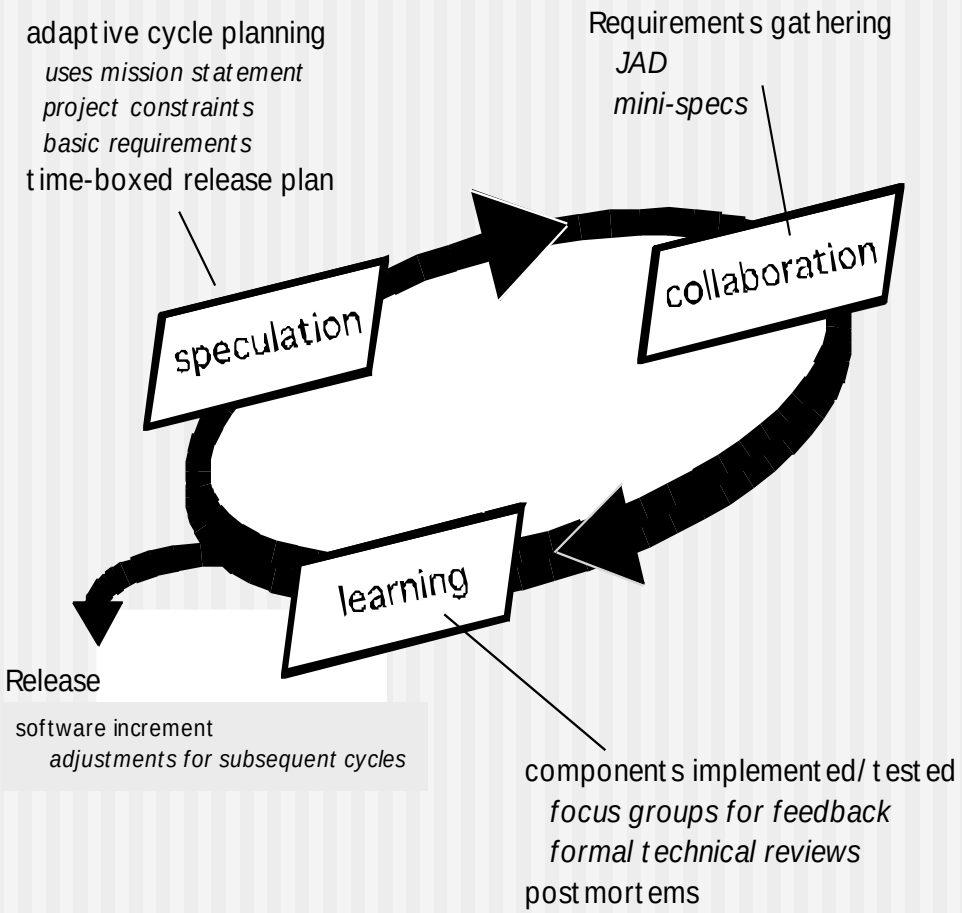
These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

توسعه نرم افزار سازگار

- توسط Jim Highsmith پیشنهاد شد
- ویژگی های متمایز ASD
  - برنامه ریزی بر اساس ماموریت
  - تمرکز بر اساس مولفه
  - استفاده از بخش بندی های مبتنی بر زمان
  - توجه صریح به ریسک ها
  - تاکید به همکاری در جمع آوری نیازمندیها
  - تاکید به یادگیری در طول فرآیند

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

توسعه نرم افزار سازگار

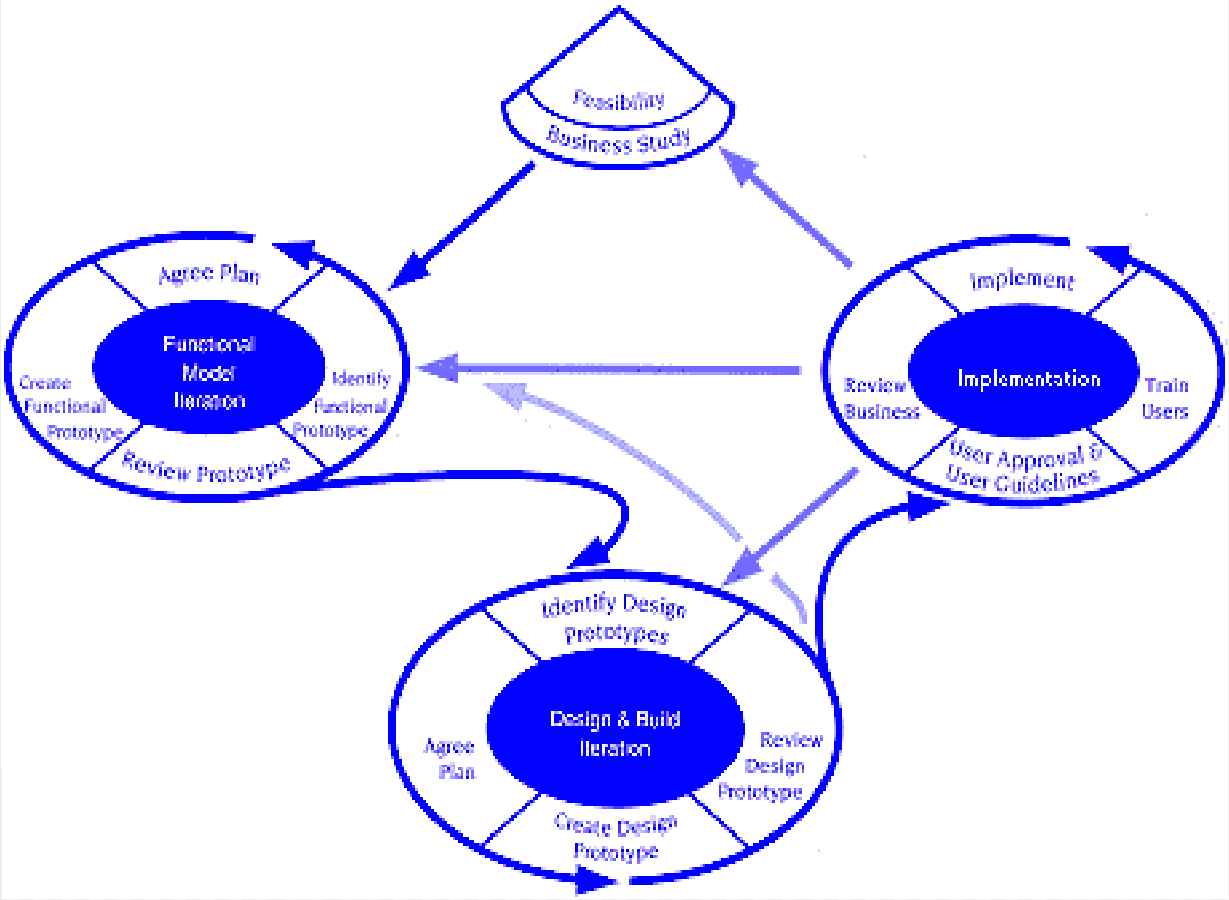


روش توسعه سیستم های پویا

- ترویج شده توسط کنسرسیوم DSDM می باشد (www.dsdm.org)
- ویژگی های متمایز DSDM
  - در بسیاری از جهات با XPو / یا ASD مشابه است.
  - اصول راهنمایی نه گانه
    - دخالت فعال کاربر ضروری است.
    - تیم DSDMباید قدرت تصمیم گیری داشته باشد.
    - تمرکز بر روی تحویل مکرر محصولات می باشد.
    - سازگاری با اهداف کسب و کار معیار اساسی برای پذیرش موارد قابل تحویل می باشد.
    - توسعه تکرار شونده و تدریجی برای تحقق یک راه حل تجاری دقیق و همگرا ضروری است .
    - همه تغییرات در طول توسعه قابل برگشت است.
    - نیازمندیها به صورت سطح بالا پایه گذاری می شوند.
    - تست در سراسر چرخه حیات مجتمع شده است.
    - رویکرد همکاری و کار اشتراکی.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

روش توسعه سیستم های پویا



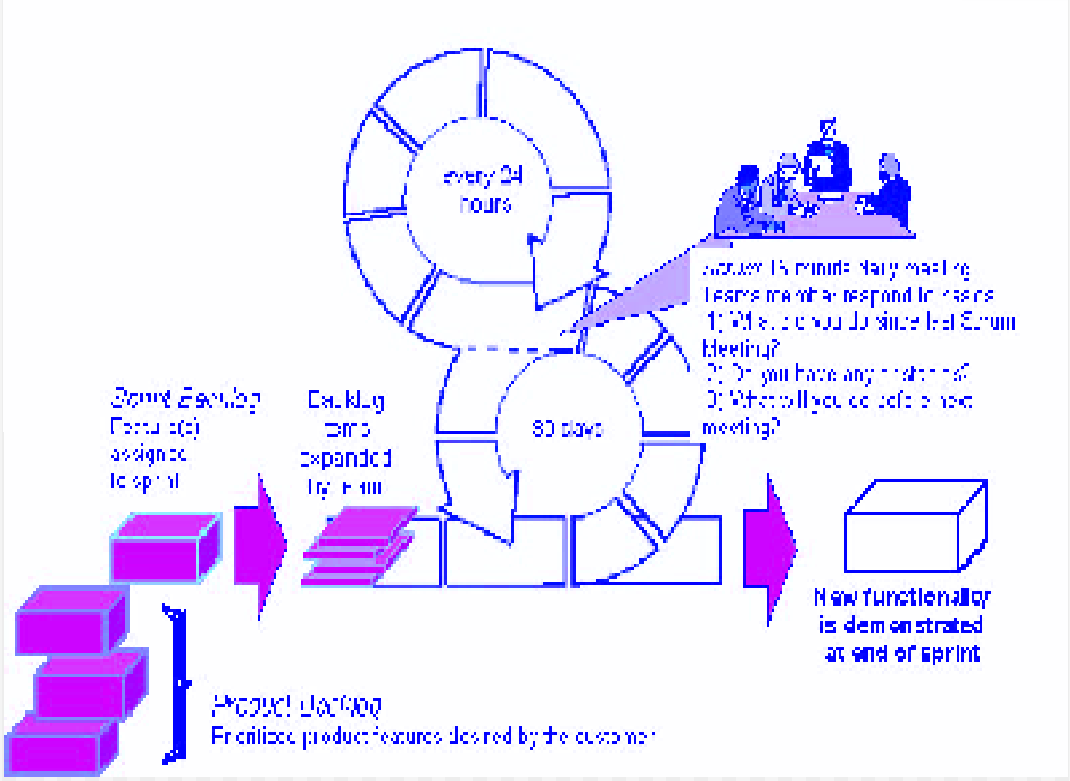
DSDM Life Cycle (with permission of the DSDM consortium)

Scrum

- توسط Schwaberو Beedle پیشنهاد شده است .
- جنبه های متمایز Scrum
  - کار توسعه به "بسته ها"تقسیم می شود.
  - آزمایش و مستندسازی در حین ساخت محصول انجام می شود.
  - کار با "مسابقه سرعت" انجام می شود و از نیازمندیهای موجود مشتق شده است.
  - جلسات بسیار کوتاه و گاهی بدون صندلی انجام می شود.
  - "دمو" ها با اختصاص زمان مشخص به مشتری تحویل داده می شوند.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Scrum



Scrum Process Flow (used with permission)

Crystal

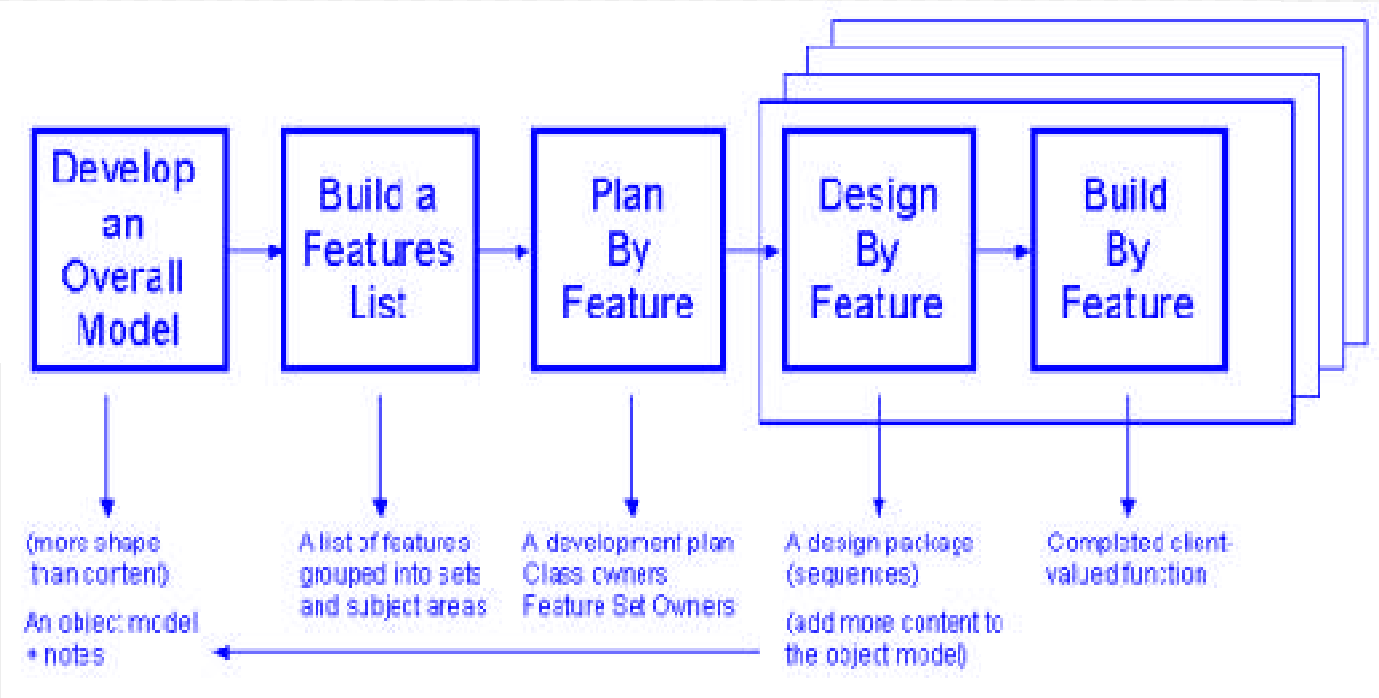
- توسط Cockburnو Highsmith پیشنهاد شده است.
- جنبه های متمایز Crystal
  - در واقع خانواده ای از مدل های فرایند است که دارای قابلیت "مانور" بر اساس ویژگی های مسئله، می باشد .
  - ارتباط چهره به چهره تاکید شده است.
  - پیشنهاد می دهد استفاده از "کارگاه های بازتاب" برای مرور کارها بعنوان عادت تیم باشد .

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

توسعه ویژگی محور

- توسط Peter Coad و همکارانش پیشنهاد شده است.
- جنبه های متمایز FDD
  - تاکید بر تعریف ویژگی ها می باشد.
  - یک ویژگی "یک عملکرد با ارزش برای مشتری است که می تواند در دو هفته یا کمتر پیاده سازی شود."
  - از یک الگوی ویژگی استفاده می شود
    - <action> the <result> <by | for | of | to> a(n) <object>
  - یک لیست از ویژگی ها ایجاد شده و "برنامه ریزی بر اساس ویژگی" انجام می شود.
  - طراحی و ساخت در FDD ادغام می شود.

توسعه ویژگی محور



Reprinted with permission of Peter Coad

مدلسازی چالاک

- توسط Scott Ambler پیشنهاد شد.
- مجموعه ای از اصول مدلسازی چالاک را پیشنهاد داد
  - مدل سازی با هدف
  - استفاده از مدل های متعدد
  - آینده نگری
  - محتوا مهم تر از نمایش
  - شناختن مدل ها و ابزارهایی که برای ایجاد آنها استفاده می کنید .
  - سازگاری محلی

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.