تجـــــزیه و تحـــلیل

مـــــهر ۱۴۰۴

فاطـمه قناویـــزی
Ghanavizi@gmail.com
@Ghanavizi
www.ghanavizi.ir

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman and Bruce R. Maxim

نرم افزار چیست؟

نرم افزار شامل:

۱- دستورالعمل ها: (یا کدها و برنامه های کامپیوتری) که هنگام اجرا ویژگی ها، عملکرد و کارایی مورد نظر را فراهم می کنند.

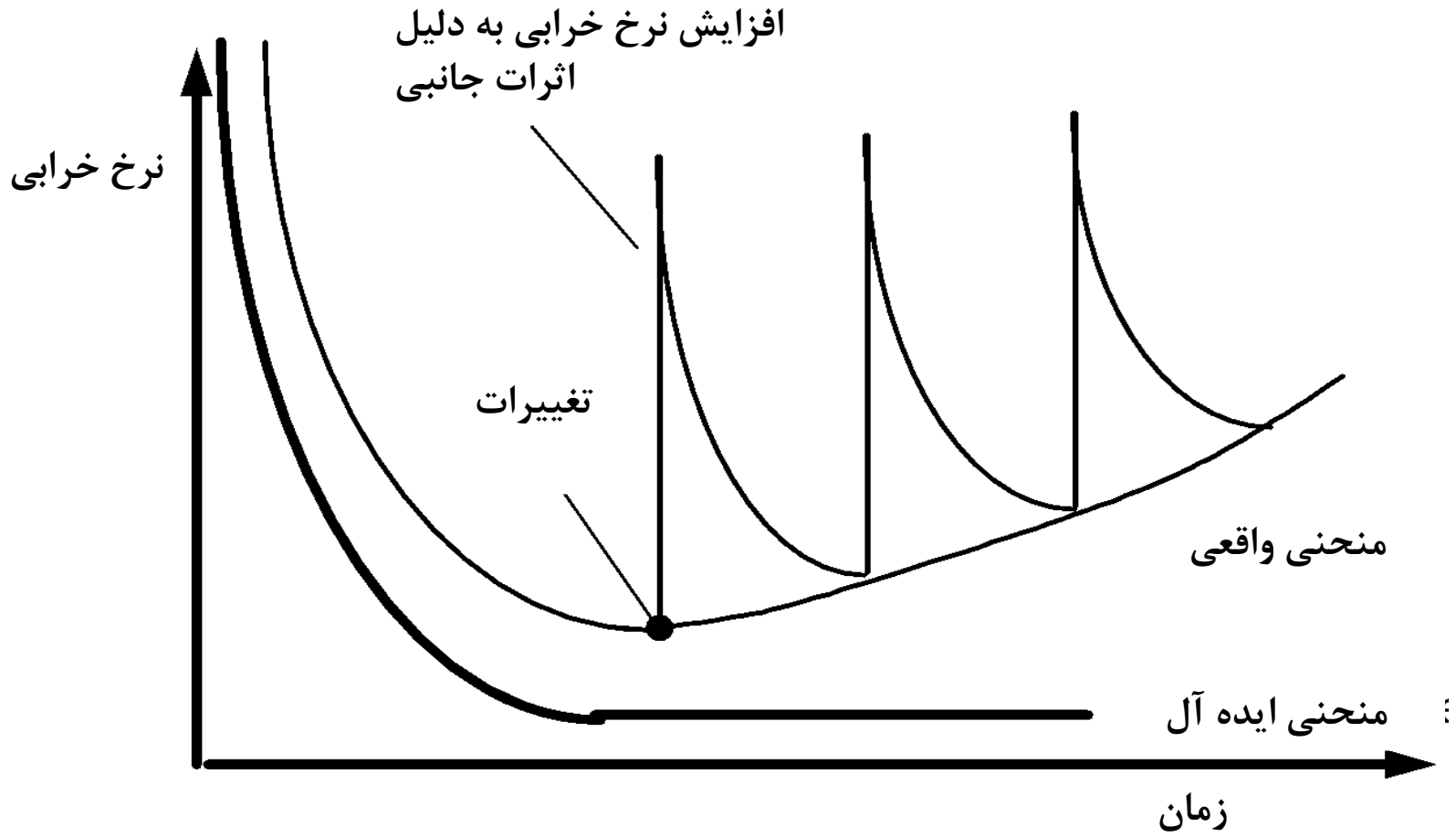
۲- ساختارهای داده: که امکان پردازش صحیح و کارآمد اطلاعات توسط برنامه‌ها را فراهم می‌کنند.(بدون داده ها، کدبی معناست. مثلا یک نرم افزار حسابداری بدون جدولهای مالی یا پایگاه داده کاربردی ندار.

۳- مستندات: که عملکرد و استفاده از برنامه ها را شرح می دهند.

نرم افزار چیست؟

- نرم‌افزار یا توسعه داده می شود یا مهندسی می شود، نه اینکه مثل یک محصول فیزیکی ساخته شود
- نرم‌افزار "فرسوده" نمی‌شود.
- اگرچه صنعت به سمت ساخت و ساز مبتنی بر مؤلفه (کامپوننت) یا ماژول های از قبل نوشته شده مثل کتابخانه ها یا افزونه ها... حرکت می‌کند، ولی بیشتر نرم‌افزارها همچنان سفارشی ساخته می‌شوند.

نمودار منحنی خرابی در مقابل منحنی ایده آل



کابردهای نـــــرم افزار

- نرم افزار سیستمی(system software): شامل سیستم‌عامل‌ها، کامپایلرها، ابزارهای مدیریت سخت‌افزار.
- نرم‌افزار کاربردی (Application software) : نرم‌افزارهایی که کاربر نهایی استفاده می‌کند مثل Word، Excel، یا اپلیکیشن‌های موبایل.
- نرم‌افزار مهندسی/علمی (Engineering/Scientific software): نرم‌افزارهای تخصصی مثل MATLAB یا نرم‌افزارهای شبیه‌سازی.
- نرم‌افزار تعبیه‌شده (Embedded software): نرم‌افزاری که در دستگاه‌ها تعبیه شده (ماشین لباسشویی، خودرو، موبایل).
- نرم‌افزار خط تولید (Product-line software): نرم‌افزارهایی که در قالب یک خانواده توسعه داده می‌شوند (مثلاً مجموعه محصولات مایکروسافت آفیس).
- برنامه‌های وب/موبایل (Web/Mobile apps): اپلیکیشن‌هایی که روی وب یا موبایل کار می‌کنند.
- نرم‌افزار هوش مصنوعی (رباتیک، شبکه‌های عصبی، بازی): نرم‌افزارهایی که بر اساس هوش مصنوعی عمل می‌کنند.

نـــرم‌افزار میـــراثی (Legacy Software)

- نرم‌افزارهای هستند که برای مدت طولانی مورد استفاده قرار گرفته‌اند و همچنان در حال استفاده هستند، اما با تکنولوژی‌های جدید و به‌روز همگام نیستند. این نرم‌افزارها معمولاً به دلایل مختلفی از جمله هزینه بالا، پیچیدگی مهاجرت، یا اهمیت و حساسیت داده‌ها، همچنان استفاده می‌شوند.
- چرا باید تغییر کنند؟
  - نرم‌افزار باید برای مواجهه با نیازهای محیط‌های جدید محاسباتی یا فناوری تطبیق(adapted) داده شود.
  - نرم‌افزار باید برای اجرای نیازمندی‌های جدید کسب‌وکار بهبود(enhanced)یابد.
  - نرم‌افزار باید برای سازگاری با سیستم‌ها یا پایگاه‌های داده مدرن‌تر گسترش(extended to make it interoperable) یابد.
  - نرم‌افزار باید برای قابل استفاده بودن در یک محیط شبکه‌ای، بازمعماری(re-architected) شود

دسته بندی دیگر نـــــرم افزاها

- **سیستم های بـــاز:** : سیستم‌هایی که با محیط خارجی تعامل دارند مثل وب سایتها.
- **سیستم های بســـته:** سیستم‌هایی که با محیط خارجی تعامل ندارند مثل برخی ماشین حسابهای قدیمی.
- **سیستم های اصلـــی و فرعـــی:** سیستم هایی فرعی سیستم های هستند که برای اهداف سیستم های اصلی تولید میشوند.

پرسش اصلی: چرا نرم‌افزار باید تغییر کند؟

- **نرم‌افزار باید:**
- با محیط‌ها و فناوری‌های جدید **سازگار شود** تا بتواند نیازهای جدید را پاسخ دهد.
- برای اجرای **نیازهای تجاری تازه، ارتقا یابد.**
- برای **قابلیت همکاری** (interoperability) با سیستم‌ها یا پایگاه‌های داده مدرن، گسترش داده شود.
- برای اینکه در یک محیط شبکه‌ای **قابل استفاده باقی بماند، دوباره معماری شود.**
- نرم‌افزار باید:
- برای محیط‌های جدید سازگار شود،
- با نیازهای جدید به‌روزرسانی گردد،
- برای تعامل با سیستم‌های مدرن گسترش یابد،
- و گاهی دوباره معماری شود تا در شبکه باقی بماند.

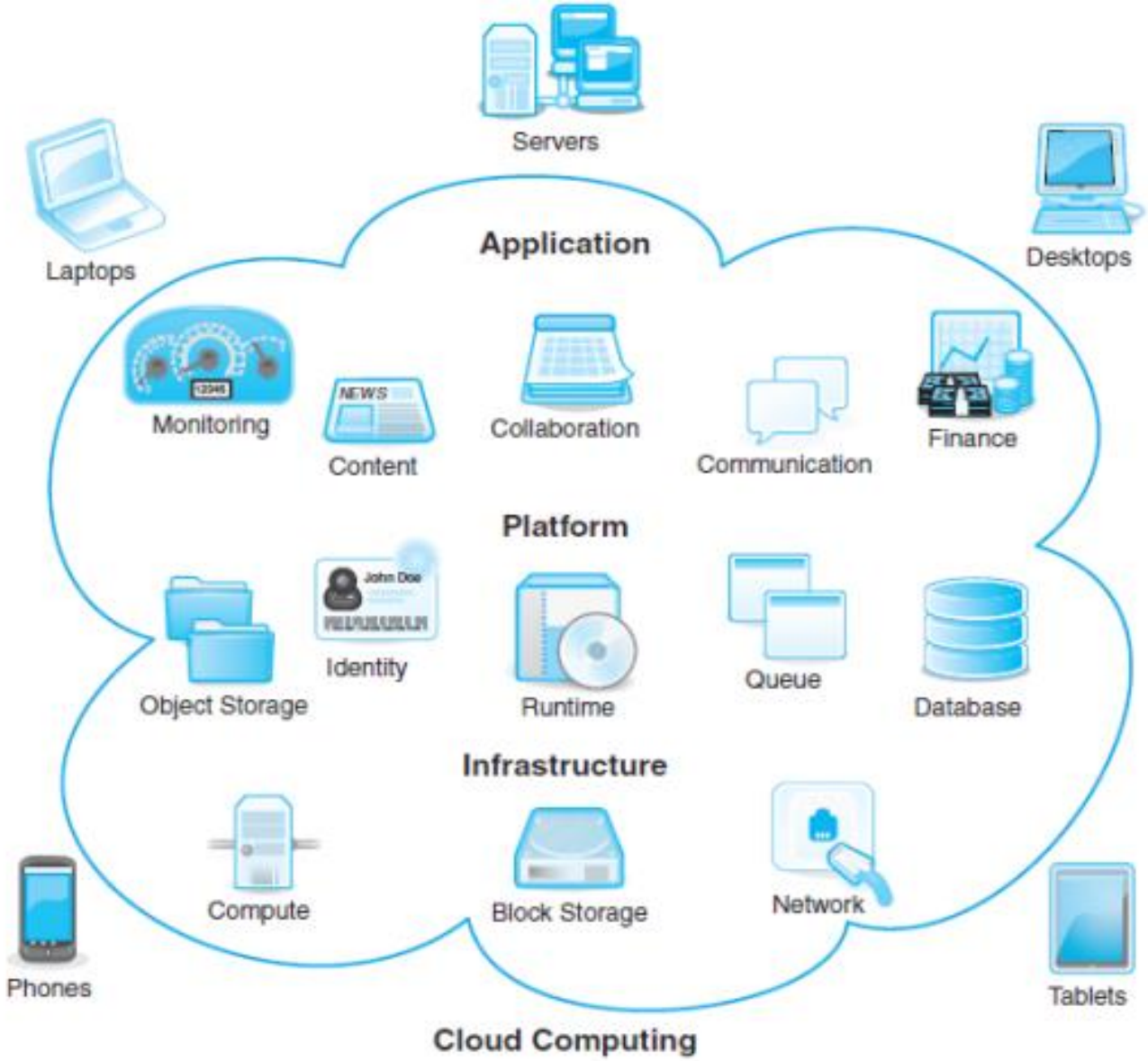
اپلیکیشنهای تحت وب (WebApps)

- ✓وب‌اپلیکیشن‌های امروزی بسیار فراتر از فایل‌های هایپرتکست (Hypertext) با چند تصویر هستند.
- ✓این برنامه‌ها با ابزارهایی مانند XML و Java تقویت می‌شوند تا به مهندسان وب امکان ایجاد قابلیت‌های محاسباتی تعاملی را بدهند.
- ✓وب‌اپلیکیشن‌ها ممکن است به کاربران نهایی قابلیت‌های مستقل ارائه دهند یا با پایگاه‌های داده و برنامه‌های تجاری شرکت‌ها یکپارچه شوند.
- ✓فناوری‌های وب معنایی (Semantic Web / Web 3.0) تکامل یافته‌اند و به برنامه‌های سازمانی و مصرفی پیچیده‌ای تبدیل شده‌اند که شامل پایگاه‌های داده معنایی، نمایش منعطف داده‌ها، و رابط‌های برنامه‌نویسی کاربردی(API ها) برای دسترسی هستند.
- ✓ماهیت زیبایی‌شناسانه‌ی محتوا (Aesthetic nature) همچنان یکی از تعیین‌کننده‌های مهم کیفیت یک وب‌اپلیکیشن است

اپلیکیشنهای موبایلی (Mobile Apps)

- ✓اپلیکیشن‌های موبایل روی پلتفرم‌هایی مانند تلفن‌های همراه یا تبلت‌ها نصب می‌شوند.
- ✓رابط کاربری آن‌ها طوری طراحی می‌شود که هم ویژگی‌های دستگاه و هم موقعیت مکانی کاربر را در نظر بگیرد.
- ✓اغلب به ترکیبی از منابع مبتنی بر وب و قابلیت‌های پردازش و ذخیره‌سازی محلی در دستگاه دسترسی دارند.
- ✓در خود پلتفرم، قابلیت ذخیره‌سازی پایدار فراهم می‌کنند.
- ✓یک اپلیکیشن وب موبایل (mobile web app) به کاربر اجازه می‌دهد تا از طریق مرورگر مخصوص موبایل، به محتوای وب دسترسی پیدا کند که متناسب با محدودیت‌ها و قابلیت‌های موبایل طراحی شده است.
- ✓یک اپلیکیشن بومی (mobile app) می‌تواند به‌طور مستقیم به سخت‌افزار دستگاه دسترسی داشته باشد تا پردازش و ذخیره‌سازی محلی را انجام دهد.
- ✓با گذر زمان، تفاوت بین این دو نوع برنامه کمتر و کمتر می‌شود

محاسبات ابری یا رایانش ابری (Cloud Computing)



این تصویر یک **نمودار مفهومی از معماری رایانش ابری** (**Cloud Computing Architecture**) است که سه لایه‌ی اصلی ابر را نشان می‌دهد:

- ✓ **زیرساخت** (**Infrastructure**)،
- ✓ **پلتفرم** (**Platform**)،
- ✓ **کاربرد** (**Application**)

- **ابر مثل یک اکوسیستم چندلایه است:**
- در **زیرساخت**، سرورها و شبکه‌ها قرار دارند
- در **پلتفرم**، محیط اجرا و ابزار توسعه وجود دارد
- در **کاربرد**، خدماتی هستند که کاربر نهایی از آن‌ها استفاده می‌کند

تمام این اجزا با هم کار می‌کنند تا ما بتوانیم هر چیزی از جمله فایل، ویدئو، یا حتی نرم‌افزار کامل را از طریق اینترنت اجرا کنیم.

رایانش ابری(Cloud Computing)

- ✓رایانش ابری منابع ذخیره‌سازی و پردازش داده‌ها را به‌صورت توزیع‌شده (Distributed) در اختیار دستگاه‌های متصل به شبکه قرار می‌دهد.
- ✓منابع پردازشی خارج از ابر قرار دارند اما به انواع منابع درون ابر دسترسی دارند.
- ✓معماری رایانش ابری شامل سرویس‌های بخش کاربر (Frontend) و بخش سرور (Backend) است.
- ✓سرویس‌های Frontend شامل دستگاه‌های کاربر و نرم‌افزارهایی است که به کاربر امکان دسترسی به منابع ابر را می‌دهند.
- ✓سرویس‌های Backend شامل سرورها، ذخیره‌سازها، و نرم‌افزارهای مقیم در سرور هستند.
- ✓معماری‌های ابری را می‌توان به گونه‌ای طراحی کرد که دسترسی به داده‌های خصوصی را محدود کنن

مقدمه و واقعیت‌های مهندسی نرم‌افزار (Some Realities)

- **برخی واقعیت‌ها:**
- باید قبل از توسعه‌ی نرم‌افزار، تلاش جدی برای **درک مسئله** انجام شود.
- **طراحی**، فعالیتی محوری و تعیین‌کننده است.
- نرم‌افزار باید **کیفیت بالایی** داشته باشد.
- نرم‌افزار باید **قابل نگهداری** باشد.

تعـــاریف مهندسی نـــرم افـــزار

تعریف اصلی (کلاسیک):

مهندسی نرم‌افزار عبارت است از به‌کارگیری اصول معتبر مهندسی برای دستیابی به نرم‌افزاری که به‌صورت اقتصادی، قابل اعتماد بوده و بر روی ماشین‌های واقعی به‌طور کارآمد عمل کند.

تعـــریف IEEE:

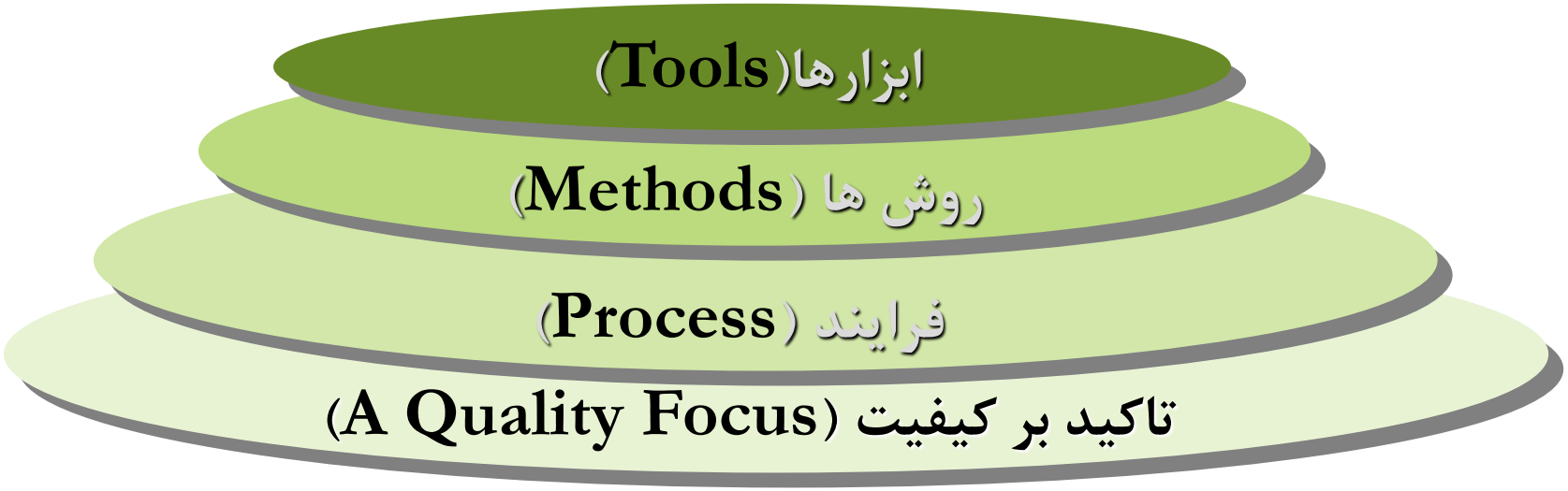
مهندسی نرم‌افزار:

۱. به‌کارگیری رویکردی سیستماتیک (نظام‌مند)، منظم و قابل اندازه‌گیری برای توسعه، اجرا و نگهداری نرم‌افزار؛ به‌عبارت دیگر، اعمال اصول مهندسی در حوزه نرم‌افزار.

۲. مطالعه روش‌ها و رویکردهایی که در بند (۱) بیان شده‌اند.

یک فناوری لایه ایی(A Layered Technology)

- A Quality Focus: تاکید بر کیفیت.
- Process: فرآیندهای تولید نرم افزار،(مثلا نیازسنجی، طراحی،پیاده سازی،آزمایش....)
- Methods: چگونگی انجام کار.(مثلا روش آبشاری، چابک...)
- Tools: ابزارها، (مثلا زبانهای برنامه نویسی، پایگاه داده های مختلف ...)



اصول کلی هوکر (Hooker's General Principles)

1: The Reason It All Exists
2: KISS (Keep It Simple, Stupid!)
3: Maintain the Vision
4: What You Produce, Others Will Consume
5: Be Open to the Future
6: Plan Ahead for Reuse
7: Think!

اصول کلی هوکر (Hooker's General Principles)

هوکر مجموعه‌ای از اصول کلی را برای مهندسی نرم‌افزار بیان کرده است:

1. دلیل وجود هر سیستم: ایجاد **ارزش برای کاربر** است.
2. **اصل KISS**: ساده نگه دار (Keep It Simple, Stupid).
3. **چشم‌انداز (Vision)**: دید کلی سیستم را همیشه حفظ کن.
4. به یاد داشته باش: محصول تو توسط دیگران مصرف خواهد شد.
5. برای **آینده باز** باش.
6. برای **استفاده مجدد (Reuse)** برنامه‌ریزی کن.
7. **فکر کن!**

افسانه‌های نرم‌افزار (Software Myths)

- ✓ افسانه‌های نرم‌افزار باورهای اشتباهی هستند که میان مدیران، مشتریان و برنامه‌نویسان رایج‌اند.
- ✓ این افسانه‌ها به دلیل داشتن بخشی از حقیقت، قابل باورند.
- ✓ اما در نهایت منجر به تصمیم‌های اشتباه و پروژه‌های شکست‌خورده می‌شوند.
- ✓ بنابراین باید با واقعیت‌های مستند جایگزین شوند.

افسانه‌های نرم‌افزار (Software Myths)

1 **افسانه‌های مدیریتی (Management Myths):**

مدیران پروژه گاهی فکر می‌کنند:

- "اگر پروژه عقب افتاد، فقط کافی است افراد بیشتری استخدام کنیم."
- "ما کتابچه‌ی دستورالعمل داریم، پس کار درست پیش می‌رود."
- "اگر پروژه را برون‌سپاری کنیم، دیگر مسئولیتی نداریم."

📘 *واقعیت:*

اضافه کردن نیرو در مراحل پایانی پروژه معمولاً باعث تأخیر بیشتر می‌شود (قانون معروف *Brooks's Law*: «اضافه کردن افراد به پروژه‌ی دیرکرده، آن را دیرتر می‌کند»).

افسانه‌های نرم‌افزار (Software Myths)

2 افسانه‌های مشتری (Customer Myths):

مشتریان اغلب می‌گویند:

- "یک بیان کلی از هدف کافی است، جزئیات را بعداً اضافه می‌کنیم."
- "اگر نیازها تغییر کرد، اصلاحش ساده است، چون نرم‌افزار انعطاف‌پذیر است."

📘 *واقعیت:*

تغییر نیازها بعد از طراحی هزینه‌برترین بخش توسعه است.

هر تغییر کوچک در ظاهر، می‌تواند معماری کل سیستم را تغییر دهد.

افسانه‌های نرم‌افزار (Software Myths)

3 افسانه‌های برنامه‌نویسان (Practitioner Myths):

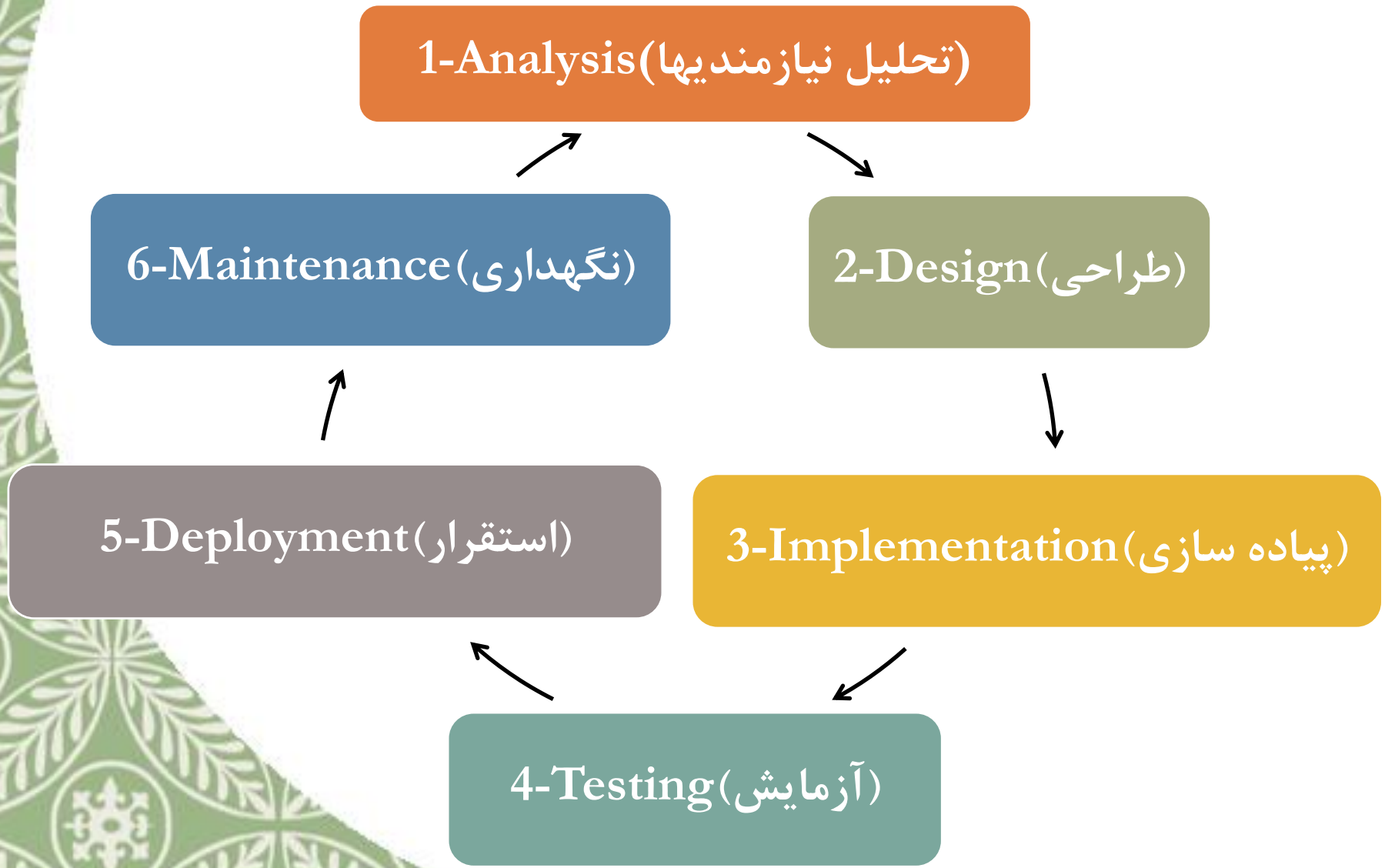
برنامه‌نویسان بعضاً باور دارند:

- "وقتی کد اجرا شد، کار تمام است."
- "تا وقتی برنامه را اجرا نکرده‌ایم، نمی‌توانیم درباره کیفیت قضاوت کنیم."
- "نوشتن مستندات فقط وقت تلف کردن است."

📘 *واقعیت:*

تحویل نرم‌افزار تازه آغاز کار است؛ نگهداری و پشتیبانی، بخش عمده‌ی عمر سیستم را تشکیل می‌دهد.

چرخه حیات پروژه های نرم افزاری
Software Development Life Cycle(SDLC)



23

چرخه حیات پروژه های نرم افزاری
(SDLC)Software Development Life Cycle

1. تحلیل نیازها :(Requirements Analysis) دراین مرحله نیازها و خواسته‌های کاربران و ذینفعان شناسایی و مستندسازی می‌شود.
2. طراحی سیستم :(System Design) بر اساس نیازهای مستندسازی شده، طراحی سیستم انجام می‌شود. این شامل طراحی معماری، طراحی دیتابیس و طراحی رابط کاربری است.
3. پیاده‌سازی :(Implementation) در این مرحله کد نویسی و توسعه نرم‌افزار انجام می‌شود. تیم توسعه بر اساس طرح‌های تهیه شده در مرحله قبل، نرم‌افزار را ایجاد می‌کند.
4. تست :(Testing) در این مرحله نرم‌افزار توسعه‌یافته تست می‌شود تا از عملکرد صحیح آن اطمینان حاصل شود. این شامل تست‌های واحد، تست‌های یکپارچگی و تست‌های سیستم است.
5. استقرار :(Deployment) پس از تست موفقیت‌آمیز، نرم‌افزار آماده استقرار و تحویل به مشتریان است. این مرحله شامل نصب، پیکربندی و راه‌اندازی نرم‌افزار در محیط عملیاتی است.
6. نگهداری :(Maintenance) پس از استقرار، نرم‌افزار نیاز به نگهداری دارد. این شامل رفع اشکالات، به‌روزرسانی‌ها و بهبودهای مداوم است.

متدولوژی های رایج در SDLC

مدلهای سنتی:

۱- مدل آبشاری (Waterfall)

۲- مدل V

مدلهای تکاملی و تکراری:

۳- مدل تکراری (Iterative)

۴- مدل تدریجی(Incremental)

۵- مدل مارپیچی یا حلزونی (Spiral)

۶- مدل نمونه سازی(Prototyping)

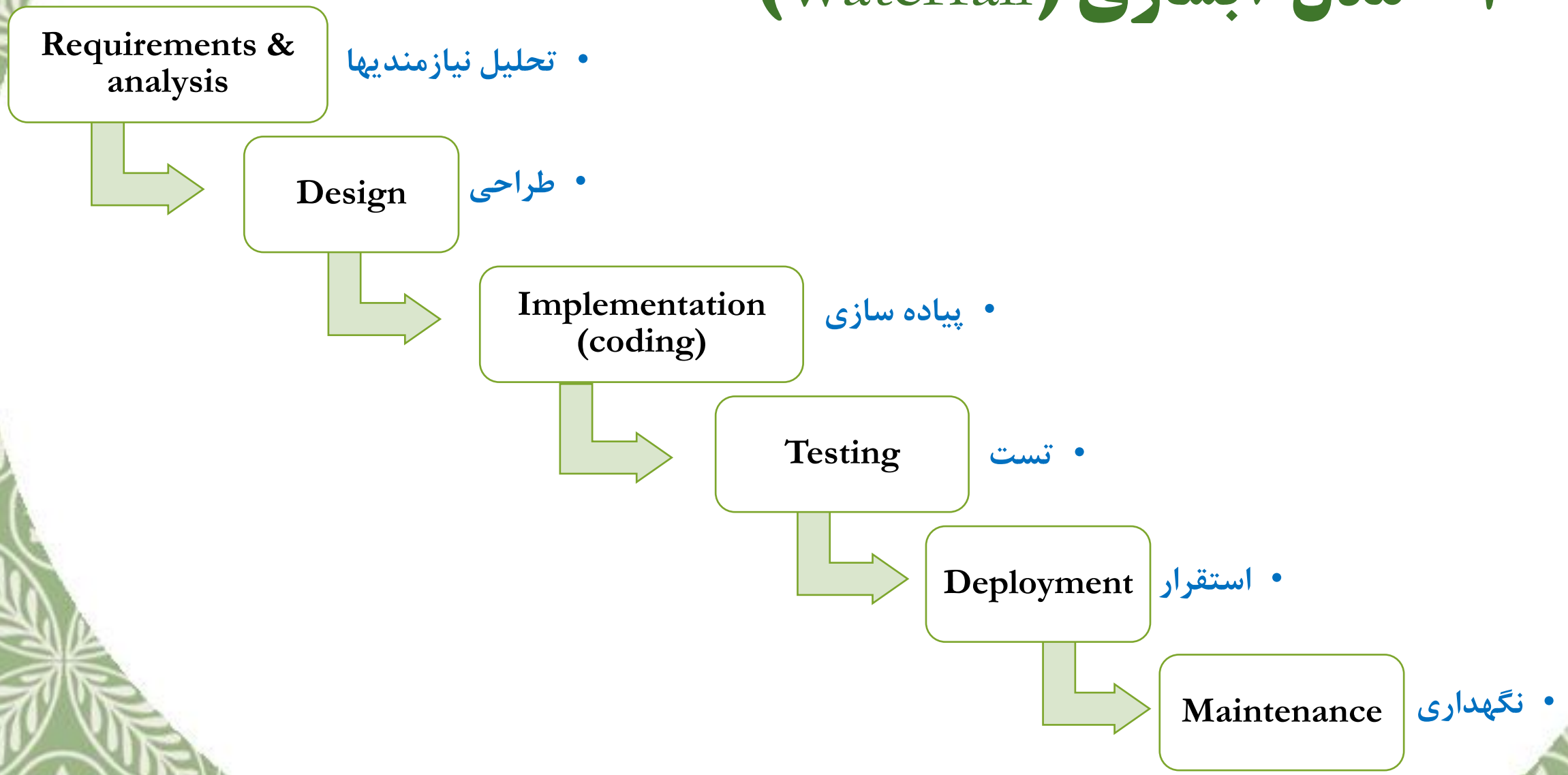
مدلهای چابک (Agile)

۷- مدل اسکرام(scrum)

۸- کانبان (Kanban)

25

۱– مدل آبشاری (Waterfall)



26

۱– مدل آبشاری (Waterfall)

یکی از قدیمی‌ترین و سنتی‌ترین مدل‌های توسعه نرم‌افزار است. این روش بر اساس یک رویکرد **خطی و متوالی** طراحی شده است، به این معنا که هر مرحله از توسعه باید قبل از شروع مرحله بعد به طور کامل تکمیل شود.

مزایا

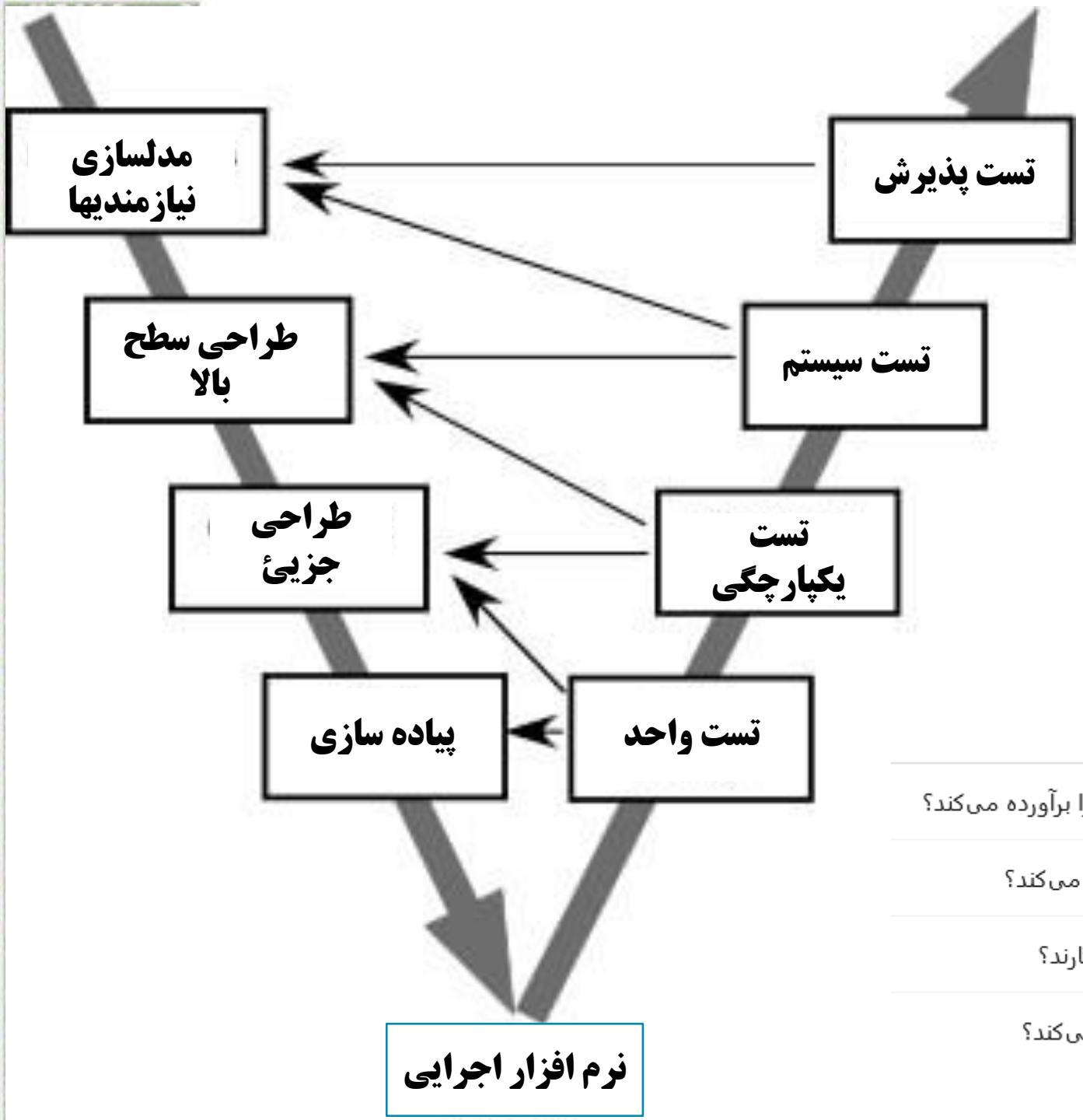
1. ساده و قابل فهم
2. برنامه‌ریزی و زمان‌بندی دقیق
3. مستندسازی کامل

معایب

1. انعطاف‌پذیری کم
2. زمان‌بر
3. خطاها دیر شناسایی می‌شوند
4. نیاز به پیش‌بینی دقیق

27

۲- مدل V



فاز طراحی	فاز تست متناظر	هدف
تحلیل نیازمندی‌ها	تست پذیرش کاربر	آیا محصول انتظارات کاربر را برآورده می‌کند؟
طراحی سطح بالا	تست سیستم	آیا کل سیستم به‌درستی کار می‌کند؟
طراحی جزئی	تست ادغام	آیا اجزای سیستم با هم سازگارند؟
پیاده‌سازی	تست واحد	آیا هر بخش کد درست کار می‌کند؟

۲– مدل V

مدلV یک رویکرد ساختاریافته برای توسعه و تست پروژه‌ها است که روی تست در مراحل مختلف توسعه تاکید دارد. در هر مرحله از مدل وی، یک تست مرتبط وجود دارد تا اطمینان حاصل شود که همه چیز همانطور که در نظر گرفته شده، عمل می‌کند. مدل وی مشابه مدل آبشاری است که در آن تسک‌ها به ترتیب و گام به گام سازماندهی می‌شوند. با این تفاوت که در مدل وی، به جای یک خط مستقیم، از یک نمودار V شکل استفاده می‌کنیم.

مزایا

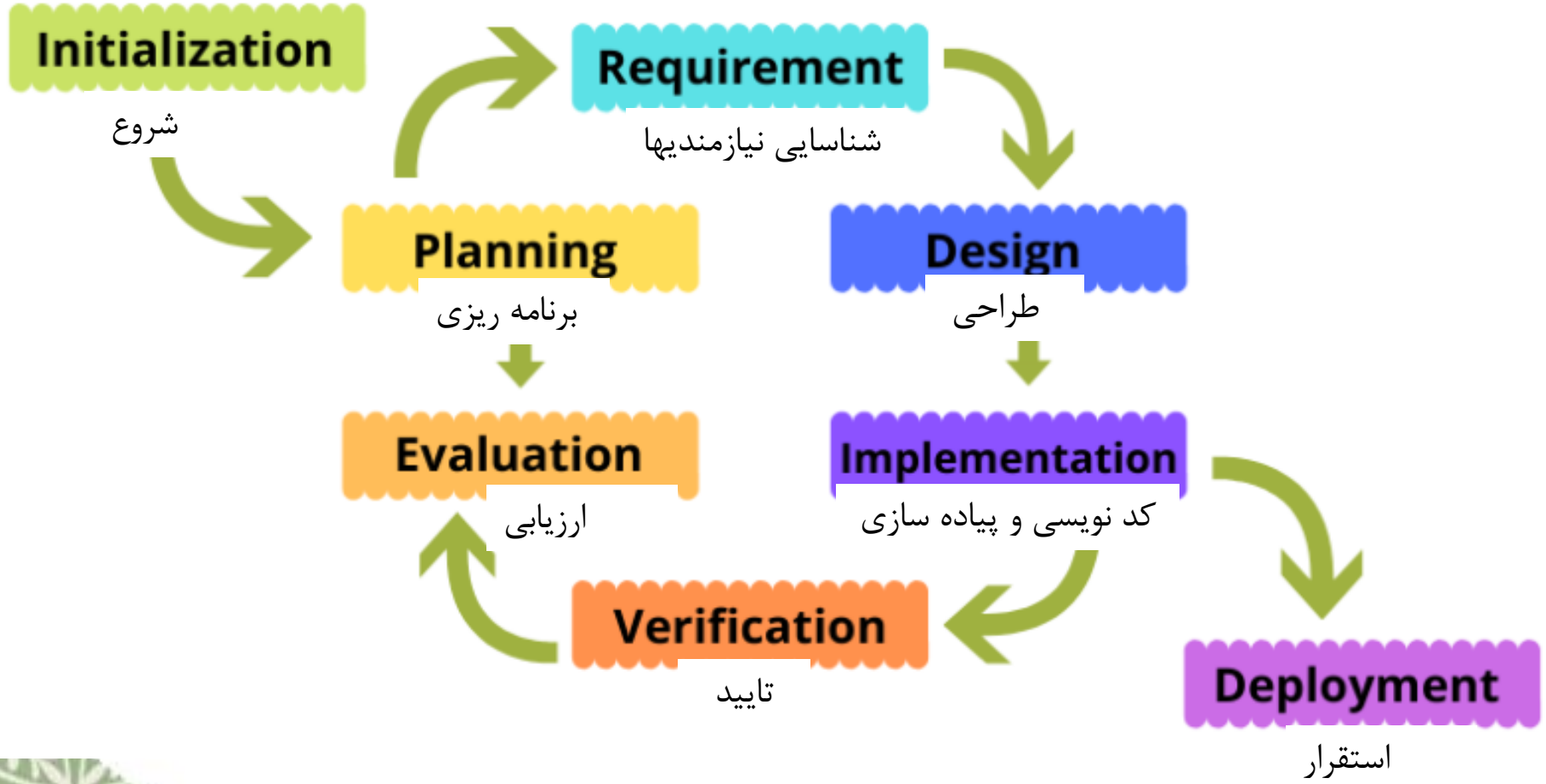
1. **تمرکز بر تست و تضمین کیفیت**
2. **کاهش هزینه‌ها:** با شناسایی مشکلات در مراحل اولیه، هزینه‌های اصلاح و رفع اشکالات کاهش می‌یابد.
3. **مستندسازی کامل**

معایب

1. **انعطاف‌پذیری کم در برابر تغییرات**
2. **زمان‌بر:** مانند مدل آبشاری، تکمیل هر مرحله نیازمند زمان زیادی است و تا تکمیل تمام مراحل، نرم‌افزار قابل استفاده نیست.
3. **مشکلات در شناسایی نیازها**
4. **نیاز به پیش‌بینی دقیق**

29

۳– مدل تکراری(Iterative)



30

۳– مدل تکراری(Iterative)

فرآیند تکرارپذیر با پیاده‌سازی ساده زیرمجموعه‌ای از نیازمندی‌های نرم‌افزار شروع می‌شود و به طور مکرر نسخه‌های در حال تکامل را تا زمانی که سیستم کامل پیاده‌سازی شود، ارتقا می‌دهد. به جای شروع با الزامات کاملاً شناخته شده، مجموعه ای از الزامات نرم افزاری را پیاده سازی می کنید، سپس الزامات بعدی را آزمایش، ارزیابی و مشخص می کنید. در هر تکرار، تغییراتی در طراحی انجام می‌شود و قابلیت‌های کاربردی جدیدی اضافه می‌شوند. برای پروژه های کوچک یا متوسط استفاده می شود.

مزایا

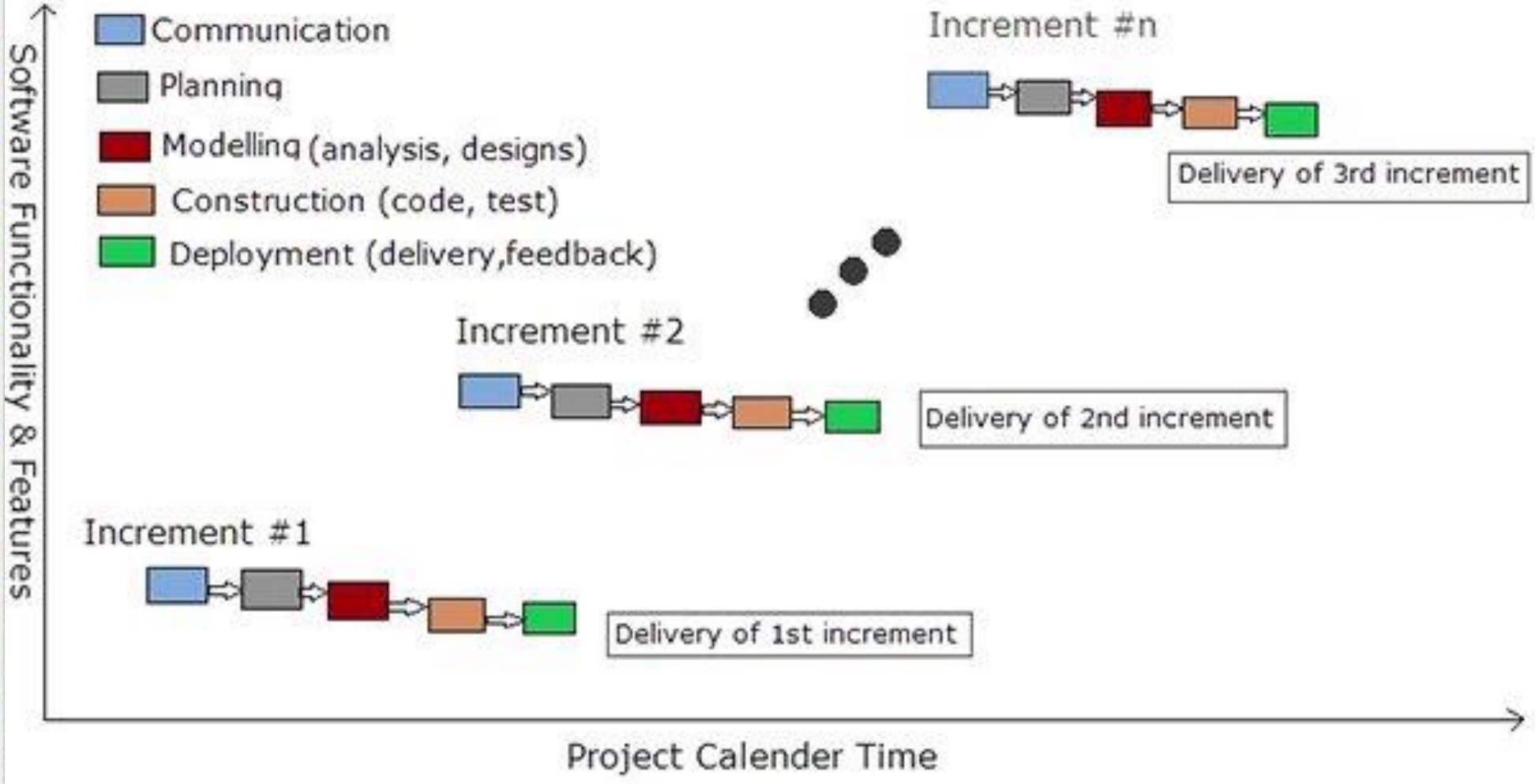
1. راه اندازی سریع پروژه
2. کاهش ریسک
3. انعطاف پذیری و اعمال اصلاحات
4. ارائه ورژن های جدید بصورت منظم
5. دریافت بازخورد مفید

معایب

1. **مشکلات احتمالی با معماری و تیم توسعه**
2. **افزایش زمان هزینه به دلیل تکراری بودن**
3. **درگیر بودن زیاد با کاربر نهایی(دریافت بازخورد مداوم)**
4. **عدم پیش‌بینی دقیق**: به دلیل تغییرات مداوم پیش بینی هزینه و زمان مشکل میشود.

31

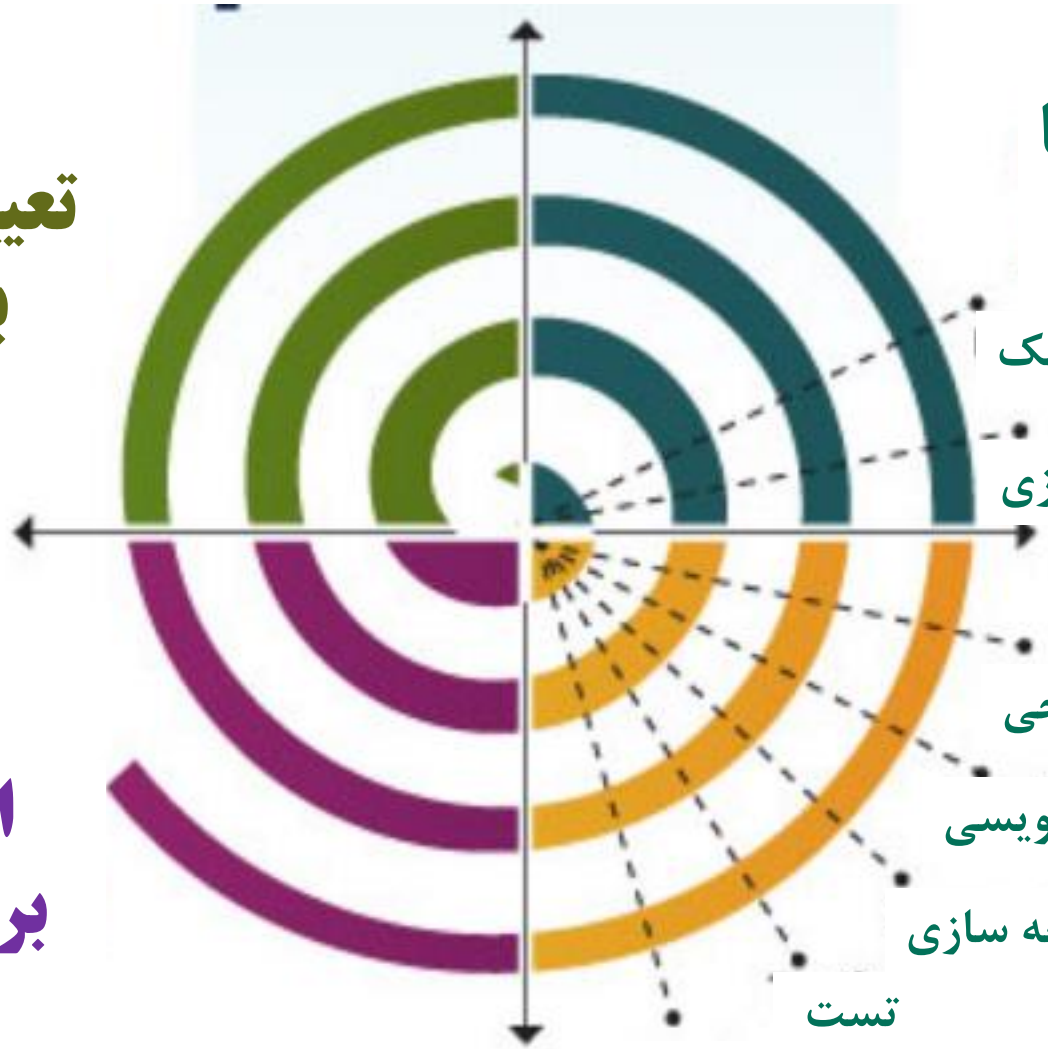
۴- مدل افزایشی (Incremental Model)



این مدل‌ها پروژه را به بخش‌های کوچکتری تقسیم می‌کنند که هر کدام شامل یک تکرار است. هر تکرار یک نسخه قابل اجرا از نرم‌افزار را تولید می‌کند که قابلیت‌ها و ویژگی‌های جدید را اضافه می‌کند. این مدل به تیم توسعه اجازه می‌دهد تا با هر تکرار بازخورد دریافت کرده و تغییرات لازم را اعمال کند.

32

۵– مدل مارپیچی یا حلزونی(spiral)



33

۵– مدل مارپیچی یا حلزونی(spiral)

به طور خاص بر ارزیابی و مدیریت ریسک‌ها تمرکز دارد. هر تکرار با نگاه کردن به ریسک های احتمالی و کشف بهترین روش برای جلوگیری یا کاهش آنها شروع می شود. معمولا برای پروژه های بزرگ از این روش استفاده میشود.

مزایا

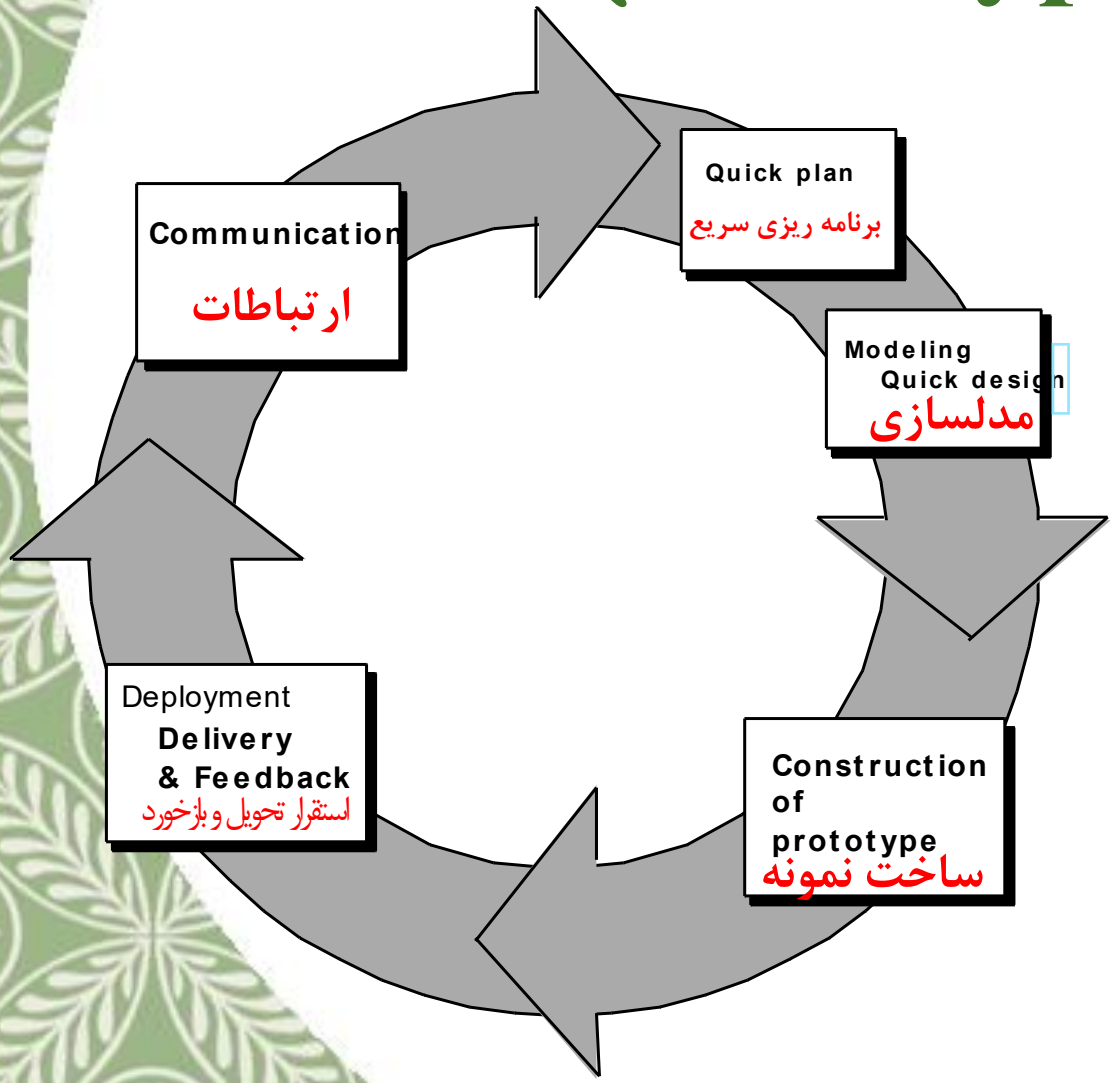
1. مدیریت ریسک
2. انعطاف‌پذیری
3. بازخورد مستمر
4. توسعه تدریجی
5. مناسب برای پروژه‌های بزرگ و پیچیده

معایب

1. پیچیدگی
2. هزینه و زمان‌بری
3. مدیریت دشوار
4. نیاز به دانش و تجربه(در زمینه ی ریسک)

34

۶– مدل نمونه سازی(Prototyping)



مزایا

1. توضیح بهتر نیازمندی‌ها
2. کاهش ریسک
3. بهبود ارتباطات با کاربر
4. افزایش رضایت مشتری

معایب

1. هزینه بر
2. احتمال افزایش زمان توسعه
3. خطر اشتباه گرفتن نمونه با محصول نهایی

35

مدل چابک (Agile)

- اجایل یک روش مدیریتی است که به تیم‌های توسعه کمک می‌کند تا محصولاتی با کیفیت بالا تولید کنند و در عین حال از انعطاف‌پذیری و تعامل مستمر با مشتری بهره ببرند.

- در واقع، Agile یک فریم‌ورک است که مجموعه‌ای از روش‌ها و فرآیندها را در بر می‌گیرد و اساس آن بر سه اصل اصلی استوار است: ۱- همکاری، ۲- تحویل مداوم ۳- بازخورد سریع.

.

مدل چابک (Agile)

چهار ارزش اصلی توسعه چابک :

1. اشخاص و تعاملات برتراز فرایندها و ابزارها
2. مشارکت مشتری برتراز قرارداد کار
3. نرم افزار قابل استفاده برتراز مستندات جامع
4. پاسخ به تغییرات برتراز دنبال کردن یک برنامه

با وجود این که اقلام سمت چپ نیز دارای ارزش هستند، اما ارزش بیشتری را برای اقلام سمت راست قائل هستیم.

37

دوازده اصل توسعه ی چابک

1. **رضایت مشتری:** بالاترین الویت است که از طریق تحویل به موقع و مداوم محصول (نرم افزار) محقق خواهد شد.
2. **استقبال از تغییر:** تغییرات در پروژه های نرم افزاری اجتناب ناپذیر هستند، حتی تغییراتی که در اواخر کار توسط مشتری درخواست شده است نیز از آنها استقبال می شود.
3. ترجیح بر آن است که در تفکر چابک، محصولات **در بازه های زمانی کوتاه (بین چند هفته تا چند ماه) و بصورت مکرر، به مشتری تحویل داده شود** (رویکرد Iterativeو Incremental)
4. **همکاری و مشارکت ذینفعان** باید در طول چرخه حیات پروژه (بصورت روزانه) وجود داشته باشد
5. **انگیزه**: پروژه ها را با استفاده از افراد با انگیزه بسازید، به اعضای تیم خود اعتماد کنید تا احساس مسئولیت پذیری داشته باشند تا کارها به خوبی انجام شود.
6. **گفتگوی رو در رو،** موثرترین روش انتقال اطلاعات به تیم پروژه می باشد
7. در تفکر چابک، تحویل محصول (نرم افزار) کاربردی به مشتری، عامل نهایی است که پیشرفت پروژه را اندازه گیری می کند.
8. **حفظ توسعه پایدار**: فرآیندهای چابک باید به سمت توسعه پایدار حرکت کنند بطوری که ذینعان تجاری،حامیان و کاربران قادر باشند سرعت پیشرفت را با روند **ثابت در طول چرخه حیات پروژه حفظ کنند**
9. **نظارت و توجه منظم باعث** افزایش چابکی شده و به برتری فنی و طراحی خوب محصول منجر خواهد شد
10. **سادگی یک ضرورت است، سادگی یعنی هنر به حداکثر رساندن کارهای انجام نشده!**
11. **تیم های چابک، خود هدایت و سازماندهی می شوند** و نیازی به گفتن آنچه باید انجام شود، نیست.
12. **لازم است تیم پروژه در فواصل منظم، کارهای خود را بررسی کند** و اگر روشی برای موثرتر شدن و پیشبرد سریع تر پروژه وجود دارد، رفتار خود را مطابق آن هم سو کند، که این اصل یکی از حیاتی ترین اصول تفکر چابک می باشد

چارچوب اسکرام(Scrum Framework)

Scrum یک چارچوب توسعه چابک است که برای مدیریت پروژه‌های پیچیده، به‌ویژه در زمینه‌ی توسعه نرم‌افزار طراحی شده.

ویژگی‌های کلیدی اسکرام:

- توسعه تدریجی و افزایشی(Incremental)
- تمرکز بر تیم‌های خودسازمان‌یافته
- چرخه‌های کوتاه توسعه به نام Sprint
- ارزش‌آفرینی سریع برای مشتری

39

چارچوب اسکرام(Scrum Framework)

40

نقش‌ها در Scrum

مالک محصول(Product Owner):

- مسئول تعریف نیازمندی‌ها در قالب Product Backlog
- مشخص می‌کند که چه چیزی باید ساخته شود
- اولویت‌بندی نیازها بر اساس ارزش تجاری
- ارتباط دائم با مشتری و ذینفعان

استاد اسکرام(Scrum Master):

- تسهیل‌گر فرآیند اسکرام
- آموزش تیم و سازمان در مورد Scrum
- حذف موانع سر راه تیم توسعه
- محافظ تیم در برابر فشارهای بیرونی

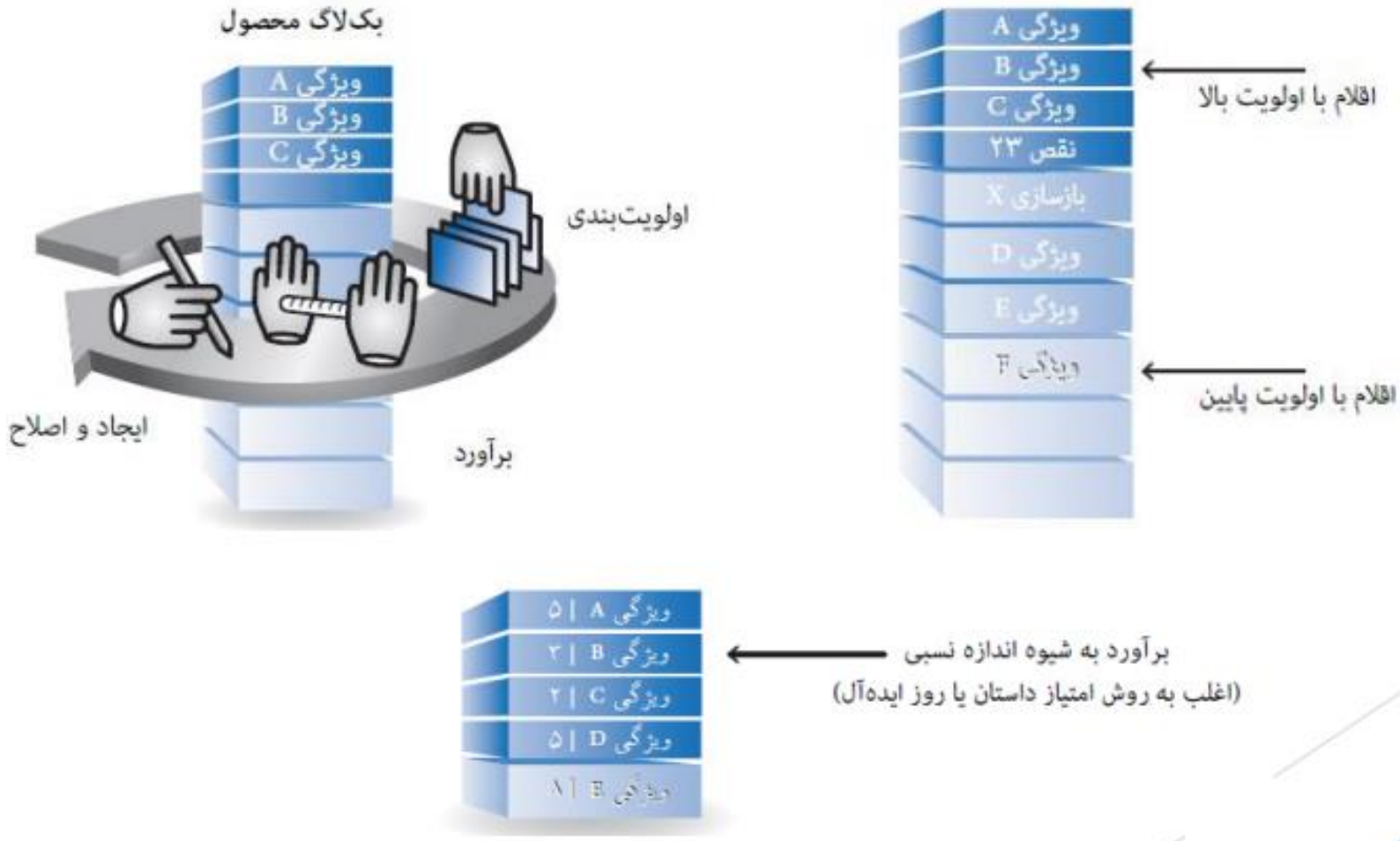
تیم توسعه(Develompment Team):

- تیمی خودسازمان‌یافته (Self-Organized)
- دارای همه مهارت‌های لازم برای تحویل محصول
- هیچ نقش داخلی رسمی مثل طراح یا تستر ندارد – همه برابرند
- برنامه ریزی اسپرینت- اجرای اسپرینت

41

Product Backlog

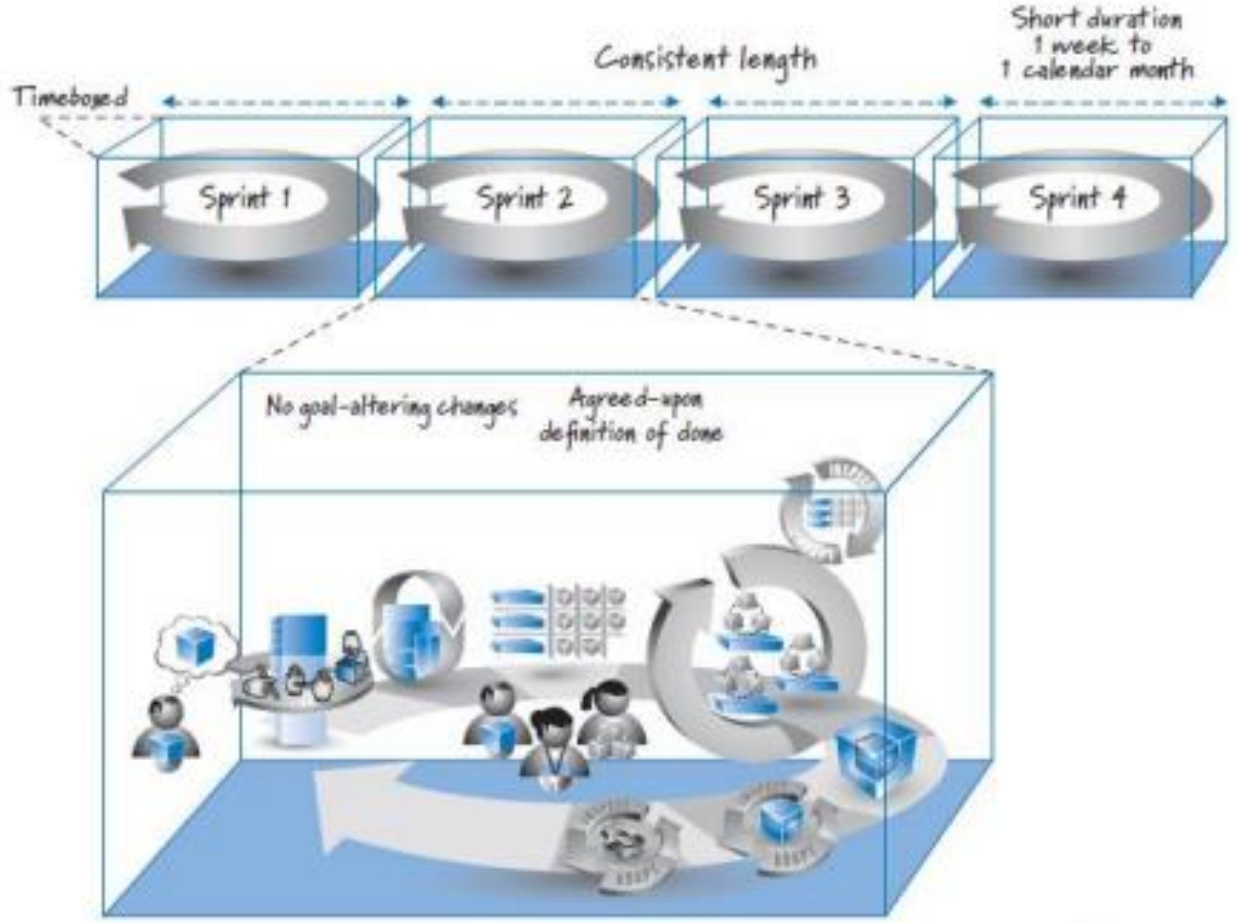
بک لاگ محصول محلی است که کلیه نیازمندی های محصول لیست شده است و اولویت بندی می شود.



42

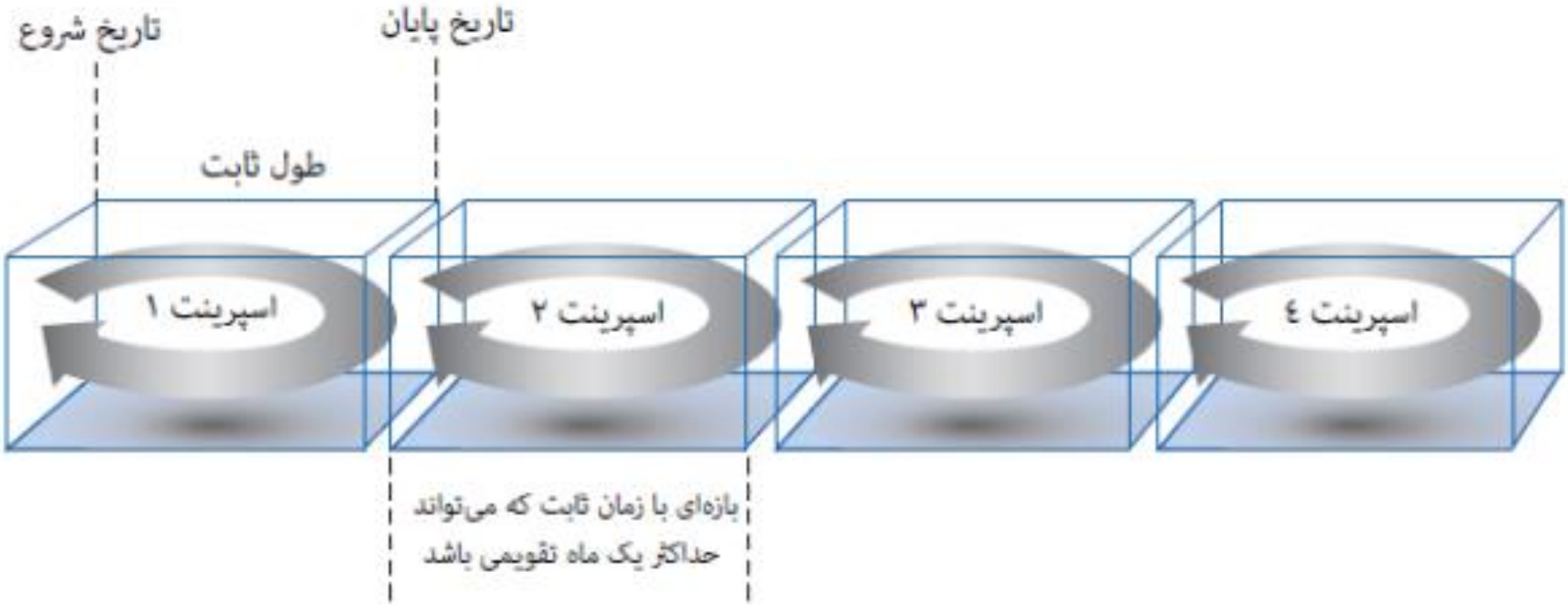
اسپرینت چیست؟

اسکرام کارها را در قالب تکرارها یا چرخه‌هایی به نام اسپرینت سازماندهی می‌کند که مدت آن‌ها حداکثر یک ماه تقویمی است.



43

اسپرینت


تاریخ شروع
تاریخ پایان
طول ثابت
اسپرینت ۱
اسپرینت ۲
اسپرینت ۳
اسپرینت ٤
بازه‌ای با زمان ثابت که می‌تواند
حداکثر یک ماه تقویمی باشد

44

رویدادهای اسکرام(Scrum Events)

۱- اسپرینت (Sprint):

- چرخه‌ای ثابت از زمان (بین ۱ تا ۴ هفته)
- همه فعالیت‌های توسعه در این بازه انجام می‌شوند
- در پایان Sprint باید یک محصول قابل تحویل آماده باشد.(Increment)

۲- برنامه ریزی اسپرینت (Sprint Planning):

- شروع هر Sprint با این جلسه انجام می‌شود.
- تیم تصمیم می‌گیرد چه کاری وچطور در Sprint انجام بشود.

سوالات اصلی: چه چیزی باید تحویل داده شود؟ چگونه انجام شود؟

۳- اسکرام روزانه (Daily Scrum):

- جلسه ایی ۱۵ دقیقه ایی هر روز فقط برای تیم توسعه. با هدف هماهنگی سریع بین اعضای تیم
- پرسشهای کلیدی: دیروز چکارکردم، امروز چکار میکنم؟ چه موانعی دارم؟

۴- بازبینی اسپرینت (Sprint review):

- درپایان هر اسپرینت نمایش محصول ساخته شده به ذینفعان
- دریافت بازخورد و بررسی تغییرات مورد نیاز.

۵- باز اندیشی اسپرینت (Sprint Retrospective)

- بررسی فرایند و عملکرد تیم
- تشخصی نکات مثبت و منفی
- برنامه ریزی برای بهبود عملکرد در اسپرینت بعدی

مصنوعات اسکرام (Scrum Artifacts)

1. Product Backlog(بک لاگ محصول)
   - فهرستی اولویت‌بندی‌شده از ویژگی‌ها، اصلاحات، وظایف و نیازمندی‌ها.
   - زنده است؛ یعنی همیشه قابل به‌روزرسانی‌ست.
   - توسط مالک محصول آماده سازی میشود.
2. Sprint Backlog (بک لاگ اسپرینت)
   - زیرمجموعه‌ای از بک لاگ محصول اس که در یک اسپرینت خاص قرار میگیرد.
   - شامل آیتم‌هایی است که تیم تصمیم گرفته در اسپرینت جاری تحویل دهد.
3. Increment
   - خروجی واقعی اسپرینت.
   - باید قابل استفاده، تست‌شده و با کیفیت باشد.
   - هر Increment، نسخه‌ای قابل تحویل از محصول است.

46

داستان کاربر(User Story)

۱- ساختار یک داستان کاربر

معمولاً یک استوری به این فرمت نوشته می‌شود:
"به عنوان [نقش کاربر]، می‌خواهم [هدف] تا [مزیت]."
مثال:
"به عنوان کاربر، می‌خواهم بتوانم با ایمیل و رمزعبور وارد سیستم شوم تا به اطلاعات شخصی خود دسترسی داشته باشم."

۲- ویژگی‌های یک استوری خوب (INVEST)

Independent مستقل: نباید به استوری دیگری وابسته باشد.

Negotiable قابل مذاکره: جزئیات آن در گفتگو بین تیم و مالک محصول مشخص می‌شود.

Valuable باارزش: باید برای کاربر یا کسب‌وکار ارزش ایجاد کند.

Estimable قابل تخمین: تیم باید بتواند اندازه و پیچیدگی آن را تخمین بزند.

Small کوچک: باید در یک اسپرینت قابل انجام باشد (معمولاً بین ۱ تا ۳ روز کاری).

Testable قابل تست: باید معیارهای پذیرش (Acceptance Criteria) داشته باشد تا بتوان آن را تست کرد.

داستان کاربر

۳- اجزای یک استوری در اسکرام

- عنوان : توصیف کوتاه از عملکرد (مثلاً "ورود به سیستم با ایمیل").
- توضیح : همان ساختار استاندارد داستان کاربری.
- معیارهای پذیرش (Acceptance Criteria) : شرایطی که استوری باید داشته باشد تا "تکمیل شده" محسوب شود.
  اولویت (Priority): تعیین می‌کند که کدام استوری زودتر انجام شود (مثلاً بر اساس ارزش کسب‌وکار).
- تخمین (Estimation): معمولاً با امتیاز داستان (Story Points) یا ساعت کاری تخمین زده می‌شود.

۴- مراحل استوری در اسکرام

- ایده‌پردازی: جمع‌آوری نیازها از ذینفعان.
- نوشتن استوری: توسط مالک محصول.
- پالایش (Refinement) تیم استوری‌ها را بررسی و جزئیات را مشخص می‌کند.
- برنامه‌ریزی اسپرینت (Sprint Planning): انتخاب استوری‌ها برای اسپرینت جاری.
- توسعه و تست: پیاده‌سازی استوری توسط تیم.
- مرور(Review): نمایش نتیجه به ذینفعان.
- بازنگری (Retrospective): بهبود فرآیند برای اسپرینت بعدی.

48

Definition of Done (DoD)
تعریف انجام شده ها

1. تعریف Done، یک معیار و نشانه ی تیم است برای مشخص‌کردن اینکه "آیا یک کار کامل شده یا نه؟" پس اگر اعضای تیم یک تسک یا فعالیت را برچسب done گذاشتند یعنی مراحل زیر انجام شده است:
2. مثال‌هایی از شرایط Done:
   - کد نوشته و تست شده باشد
   - مستندات به‌روزرسانی شده باشد
   - تأیید نهایی توسط تیم QA
   - بدون باگ‌های حیاتی

49

ارزش‌های اسکرام

1. تمرکز (Focus) : تمرکز بر اهداف Sprint
2. شجاعت (Courage) : بیان واقعیت، پذیرش چالش‌ها
3. تعهد (Commitment) : پایبندی به اهداف تیمی
4. احترام(Respect) : احترام متقابل بین اعضای تیم
5. شفافیت (Openness) : اشتراک‌گذاری اطلاعات و موانع

50

کانبان چیست؟

- کانبان یکی از روشهای اجرای مدل چابک در توسعه ی نرم افزار است. که جریان کار را مدیریت میکند و از تابلو و کارتهای دیداری استفاده میکند.

- تمرکز روی:
  - ✔ مشاهده وضعیت کارها
  - ✔ محدود کردن کارهای در حال انجام
  - ✔ بهبود مداوم فرآیندها

51

اجزای اصلی کانبان

1. تابلو یا Board : ساختار دیداری
2. ستون ها یا Columns:
   - برای انجام (To Do)
   - در حال انجام (In Progress)
   - انجام شده (Done)
3. کارتها یا cards: هر تسک روی یک کارت نوشته میشود.
4. محدودیت تعداد کارهای در حال اجرا یا WIP Limit (Work In Progress)
5. خطوط مسئولیت (Swimlanes) این‌ها خطوطی افقی هستند که بُرد را به بخش‌های مختلف تقسیم می‌کنند.

52

مقایسه

ویژگی	سنتی	اسکرام	کانبان
ساختار زمانی	وابسته به پروژه	اسپرینت	بدون زمان ثابت
نقش‌ها	مدیر پروژه	رسمی و ثابت	آزاد و انعطاف‌پذیر
تغییر در جریان	نیازمند مجوز	سخت	آسان و لحظه‌ای

53

مزایای کانبان

- ساده و قابل درک برای همه
- انعطاف‌پذیر در برابر تغییر
- تمرکز بر تحویل مداوم
- جلوگیری از شلوغی وظایف
- قابل استفاده در تیم‌های بزرگ، کوچک و حتی فردی

54

مفهوم سیستم و تحلیل سیستم

- **سیستم چیست؟**
  - مجموعه‌ای از اجزا که با یکدیگر تعامل دارند و برای رسیدن به یک هدف مشترک کار می‌کنند.
  - هر سیستم شامل **ورودی، پردازش، خروجی، بازخورد و کنترل** است.
  - سیستم‌ها می‌توانند فیزیکی (مثل خط تولید) یا اطلاعاتی (مثل سیستم بانکی) باشند.
- **تحلیل سیستم چیست؟**
  - فرآیند مطالعه دقیق یک سیستم موجود به‌منظور شناخت مشکلات، نیازها، جریان اطلاعات و عملکرد واقعی آن.
- **هدف تحلیل سیستم:**
  - افزایش کارایی
  - کاهش هزینه
  - ساده‌سازی فرآیندها
  - ایجاد پایه برای طراحی سیستم جدید یا اصلاح سیستم موجود
- **چرا تحلیل سیستم مهم است؟**
  - سازمان‌ها بدون تحلیل دقیق، در طراحی سیستم‌های جدید دچار اتلاف منابع، دوباره‌کاری و شکست پروژه می‌شوند.
  - تحلیل سیستم تصمیم‌گیری مدیران را مبتنی بر داده و واقعیت می‌کند.

55

تحلیل‌گر سیستم: نقش، مهارت‌ها، خروجی‌ها

- **تحلیل‌گر سیستم کیست؟**
  - فردی که سیستم موجود را مطالعه، مدل‌سازی، عیب‌یابی و نیازهای کاربران را استخراج می‌کند.
- **مهارت‌های کلیدی تحلیل‌گر:**
  - تفکر سیستمی
  - مهارت ارتباطی قوی
  - توان تحلیل جریان اطلاعات
  - توان مدل‌سازی (UML، DFD)
  - شناخت مدیریت پروژه و کسب‌وکار
- **وظایف تحلیل‌گر در پروژه‌های واقعی:**
  - مصاحبه با کاربران
  - تحلیل مشکلات موجود
  - تعریف نیازمندی‌ها
  - ساخت مدل‌های مفهومی
  - مستندسازی سیستم
  - همکاری با طراحان و برنامه‌نویسان

تحلیل‌گر باید بفهمد:

- اطلاعات از کجا می‌آید؟
- به چه کسی منتقل می‌شود؟
- چگونه پردازش می‌شود؟
- چه داده‌هایی تکراری یا زائد است؟
- چه نقاطی باعث تأخیر، خطا یا گلوگاه می‌شود؟

این مهارت پایه‌ی ترسیم DFD و BPMN است.

تحلیل‌گر باید سیستم را به یک **نقشه قابل درک** برای برنامه‌نویس تبدیل کند.

- مدل‌سازی به معنی ساده‌سازی سیستم پیچیده است.
- خروجی مدل‌سازی شامل موارد زیر است:
  - Use Case Diagram (نیازهای کاربر)
  - Activity Diagram (فرآیندها)
  - Sequence Diagram (تعاملات زمانی)
  - DFD (جریان داده)
  - Class Diagram (ساختار نرم‌افزار)

56

تحلیل‌گر سیستم: نقش، مهارت‌ها، خروجی‌ها

- بی‌طرفی
- تمرکز بر مسئله، نه افراد
- استفاده از داده و مدل برای حل اختلاف
- جلوگیری از انحراف بحث

تحلیل‌گر باید در جلسات، نقطه مشترک داشته باشد و میان ذی‌نفعان توافق بسازد؛ حتی وقتی:

- بودجه محدود است
- زمان‌بندی سخت است
- دیدگاه‌ها متضاد هستند

این توانایی یعنی تحلیل‌گر می‌تواند راه‌حل ارائه دهد که **همه احساس برد کنند**.

کاربران معمولاً ترکیبی از:

- **نیاز واقعی** (Pain Point)
- **خواسته ایده‌آل** (Wants)
- **تخیلات تکنولوژیک** (Nice-to-have)

را بیان می‌کنند.

تحلیل‌گر باید تشخیص دهد کدام مورد:

- برای موفقیت سیستم حیاتی است؟
- اختیاری است؟
- خارج از محدوده پروژه است؟

Active Listening یعنی:

- قطع نکردن کاربر
- بازگویی گفته‌های او برای تأیید
- پرسیدن سؤال‌های پیگیری
- توجه به پیام‌های غیرکلامی

تحلیل‌گر باید بتواند با جمله‌های کوتاه، دقیق و بدون دوپهلو حرف بزند.

مثل:

❌ «سیستم باید سریع باشد.»

✔ «میانگین زمان پاسخ باید کمتر از ۲ ثانیه باشد.»

57

تحلیل‌گر سیستم: مهارت‌ها

۱) انواع رایج کاربران ناسازگار

🔸 کاربر مقاوم در برابر تغییر (Change-Resistant User)

- از سیستم جدید می‌ترسد
- تصور می‌کند شغلش تهدید می‌شود
- مدام می‌گوید «سیستم قبلی خوب بود»

🔸 کاربر بیش‌ازحد خواهان (Over-demanding User)

- همه‌چیز را می‌خواهد
- توقعات غیرواقعی دارد
- نمی‌فهمد محدودیت زمانی/بودجه وجود دارد

🔸 کاربر مبهم و غیرشفاف (Unclear User)

- نیازها را دقیق بیان نمی‌کند
- توضیحاتش متناقض است
- از بیان مشکل واقعی طفره می‌رود

🔸 کاربر سلطه‌جو (Dominant User)

- می‌خواهد مسیر تحلیل را کنترل کند
- اجازه نمی‌دهد دیگران نظر بدهند
- تحلیل را به سمت منافع شخصی می‌برد

🔸 کاربر بی‌تفاوت (Passive User)

- اطلاعات نمی‌دهد
- جلسه را جدی نمی‌گیرد
- مسئولیت خود را نمی‌پذیرد

🔥 گفتگوهای چالشی (Challenging Conversations)

این گفتگوها زمانی رخ می‌دهند که:

- نیازها مبهم باشند
- ذی‌نفعان تضاد منافع داشته باشند
- کاربر عصبی یا نارضی باشد
- تغییرات سازمانی حساسیت‌زا باشند

تحلیل‌گر باید بتواند این گفتگوها را **به مسیر حرفه‌ای** هدایت کند.

۱) نشانه‌های یک گفتگوی چالشی

- قطع کردن مکرر صحبت‌ها
- مخالفت‌های احساسی
- انتقاد از سیستم قدیمی یا مدیران
- بحث‌های شخصی
- انکار مشکل
- یا تبدیل شدن جلسه به بحث قدرت

۲) دلیل علمی ایجاد گفتگوهای چالشی

طبق مدل‌های رفتار سازمانی:

- ترس
- عدم شفافیت
- فشار کاری
- فقدان اعتماد
- ترس از ارزیابی یا کنترل

باعث می‌شود کاربران واکنش تدافعی نشان دهند.

تحلیل‌گر سیستم: نقش، مهارت‌ها، خروجی‌ها

چطور تحلیل گر با کاربران ناسازگار و گفتگوهای چالشی برخورد حرفه ایی میکند؟

1 بی‌طرفی مطلق (Neutrality)

تحلیل‌گر نباید طرف هیچ کس را بگیرد.

این ساده نیست، اما اگر شکلی از جانبداری دیده شود، کل تحلیل مخدوش می‌شود.

مثال:

«اجازه بدهید مطمئن شوم همه دیدگاه‌ها به درستی شنیده می‌شود.»

---

2 بازگرداندن گفتگو از احساس به منطق

به‌جای پاسخ‌دادن به احساسات، تحلیل‌گر بحث را به داده و نیاز واقعی می‌کشد.

مثال:

کاربر: «این سیستم جدید به درد نمی‌خورد.»

تحلیل‌گر: «می‌فهمم نگرانی دارید. اجازه بدهید مشخص کنیم کدام بخش برای کار روزانه شما چالش‌برانگیز است.»

3 پرسیدن سؤال‌های روشن‌کننده

برای کاربر مبهم یا ناراضی:

«می‌توانید یک مثال واقعی بزنید؟»

«در چه موقعیتی این مشکل را تجربه کردید؟»

«این مشکل چقدر تکرار می‌شود؟»

این سؤالات گفتگوی چالشی را *خلاق* می‌کنند.

---

4 بستن دهان کاربر سلطه‌جو بدون تنش

تحلیل‌گر باید اجازه ندهد یک نفر تحلیل را بدزدد.

روش حرفه‌ای:

«نظرتان را یادداشت کردم. حالا می‌خواهم نظر بقیه را هم بشنوم.»

این جمله احترام‌آمیز اما محکم است.

5 کاهش مقاومت در برابر تغییر

تحلیل‌گر باید **بفهمد مقاومت از کجا می‌آید:**

ترس از یادگیری؟ ترس از ارزیابی؟ ترس از تکنولوژی؟

استراتژی:

- نشان دادن منافع فردی
- ارائه نسخه اولیه (Prototype)
- آموزش مقدماتی

مثال:

کارمندان بانک در ابتدا به دستگاه Self-Service مقاومت د
تحلیل‌گر با ارائه نمونه اولیه و شبیه‌سازی، ترس آنها را کاه

---

6 تبدیل گفتگوی چالشی به مستند نیازمندی

تحلیل‌گر باید چالش را تبدیل کند به:

- نیاز
- محدودیت
- سناریو
- یا ریسک

چالش = منبع داده، نه مزاحمت.

59

تحلیل‌گر سیستم: نقش، مهارت‌ها، خروجی‌ها

مثال اینکه چطور یک تحلیل گر، اعتراض، و درد واقعی را به نیازمندی واقعی تبدیل میکند؟

🌟 یک مدل حرفه‌ای: COMPLAINT → PAIN → REQUIREMENT

Complaint (اعتراض)

«این سیستم جدید فقط کار را برای ما سخت‌تر کرده!»

Pain (درد واقعی)

«برای ثبت وضعیت بیمار باید در سه فرم مختلف اطلاعات تکراری وارد کنیم.»

Requirement (نیازمندی واقعی)

✔ «سیستم باید امکان ورود یک‌باره اطلاعات بیمار را فراهم کند و داده‌ها در همه فرم‌ها به‌طور خودکار همگام شوند.»

60

مراحل تحلیل سیستم (چرخه استاندارد)

- ۱) شناخت مسئله (Problem Identification)
  - شناسایی نقاط ضعف سیستم موجود
  - تعریف دقیق مشکل
  - مثال: صف‌های طولانی بانک ← نیاز به سیستم نوبت‌دهی
- ۲) جمع‌آوری اطلاعات
  - مصاحبه
  - مشاهده
  - پرسشنامه
  - مطالعه اسناد
  - استخراج داده‌ها
  - مثال واقعی: تحلیل‌گران Facebook برای «News Feed» ابتدا ۱۵۰ ساعت رفتار کاربران را مشاهده کردند.

۳) تحلیل فرآیندها

ترسیم DFD

ترسیم BPMN

بررسی جریان اطلاعات و فعالیت‌ها

۴) تحلیل نیازمندی‌ها

نیازهای کارکردی (چه کند؟)

نیازهای غیرکارکردی (سرعت، امنیت، دقت)

۵) ارائه مدل منطقی سیستم

معماری سطح بالا

تعریف موجودیت‌ها و تعاملات

61

روش‌های جمع‌آوری اطلاعات (با مثال‌های سازمانی)

۱) مصاحبه:	ساختارمند، نیمه‌ساختارمند، باز
۲) مشاهده مستقیم (Observation):	حضور تحلیل‌گر در محل
۳) پرسشنامه:	مناسب جمعیت بزرگ
۴) بررسی اسناد:	فرم‌ها، گزارش‌ها، دستورالعمل‌ها
۵) روش‌های نوین:	تحلیل لاگ‌های سیستم، داده‌کاوی، تحلیل رفتار مشتری

62

ساختار سازمان چیست؟ و چرا در تحلیل سیستم مهم است؟

- **تعریف ساختار سازمانی**
  - نحوه تقسیم وظایف، ارتباطات، جریان اطلاعات و سطح اختیار در یک سازمان.
- **چرا تحلیل ساختار سازمان مهم است؟**
  - تعیین اینکه اطلاعات از کجا وارد می‌شود و به کجا می‌رود
  - شناسایی تصمیم‌گیران
  - تعیین مسئولیت‌ها
  - تشخیص گلوگاه‌های کاری و اطلاعاتی
- **مثال واقعی:**
  - هنگام پیاده‌سازی سیستم HISدر بیمارستان‌ها:
  - فهم نقش پزشک، پرستار، داروخانه
  - فهم جریان نسخه ⟶ آزمایش ⟶ دارو
  - فهم سطوح دسترسی

انواع ساختارهای سازمانی + نمونه شرکت‌های جهانی

ساختار سلسله‌مراتبی ((Hierarchical

کلاسیک، دستورات از بالا به پایین — مثال: **Toyota**

ساختار ماتریسی (Matrix)

کارکنان دو مدیر دارند (وظیفه + پروژه) — مثال: **Microsoft**

ساختار شبکه‌ای (Network Structure)

تیم‌های کوچک و مستقل — مثال: **Spotify Model**

ساختار فرآیندی (Process-Oriented)

تمرکز بر جریان کار — مثال: **Amazon Fulfillment**

ساختار مسطح (Flat Structure)

سطوح مدیریتی کم — مثال: **Valve Corporation**

64

تحلیل محصول (Product Analysis)

- **تحلیل محصول چیست؟**
  - مطالعه ویژگی‌ها، عملکرد، ارزش، اجزا و نیاز مشتری نسبت به محصول.
- **شاخص‌های تحلیل محصول**
  - عملکرد
  - کیفیت
  - قابلیت استفاده
  - ارزش افزوده
  - هزینه تولید
  - تجربه کاربری

65

تحلیل فرآیند (Process Analysis)

- **تحلیل فرآیند چیست؟**
  - مطالعه فعالیت‌ها، ترتیب عملیات، جریان داده، نقش‌ها و نقاط اتلاف.
- **ابزارهای تحلیل فرآیند**
  - فلوچارت
  - BPMN
  - DFD
  - SIPOC
  - Swimlane Diagram

مثال واقعی: فرآیند تحویل Amazon

دریافت سفارش
تأیید پرداخت
پردازش در انبار
بسته‌بندی
ارسال
تحویل
دریافت بازخورد مشتری

66

مدلسازی فرآیندی Process-Oriented Modeling

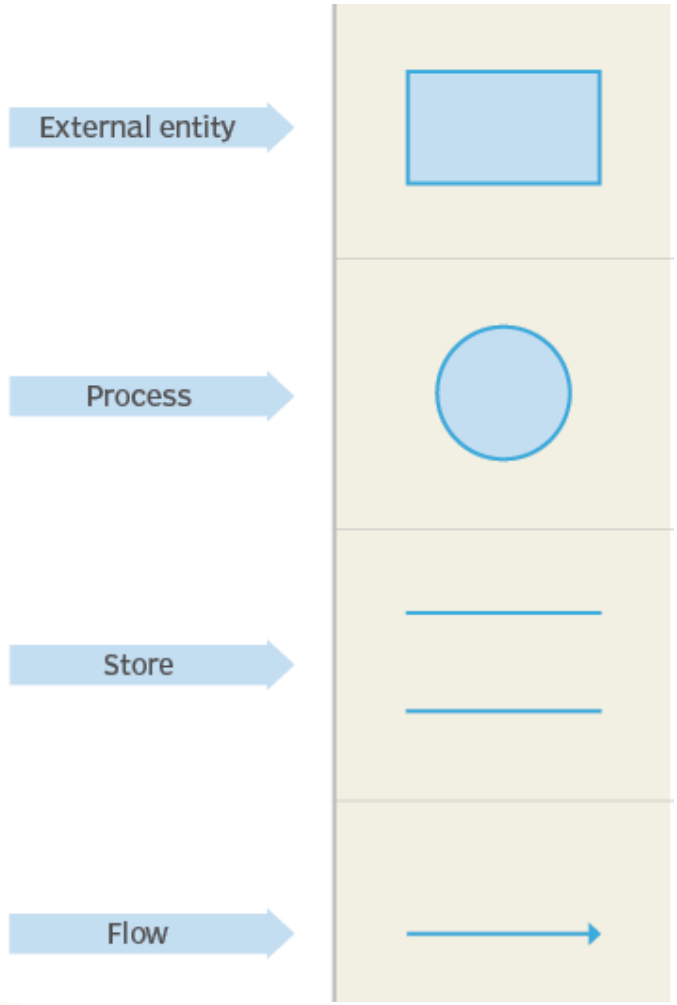
- **مدل‌سازی فرآیندی (یا مدل‌سازی سنتی) روشی برای نمایش جریان اطلاعات و داده‌ها بین اجزای مختلف یک سیستم اطلاعاتی است. در این روش تمرکز بر موارد زیر است:**
  - چه **داده‌هایی وارد سیستم میشوند.**
  - چطور این داده‌ها **پردازش میشوند.**
  - چه **خروجی‌هایی** تولید میشوند.

نمودار جریان داده یا DFD (Data flow diagram)

- **یکی از ابزاری های مدلسازی فرآیندی، نمودار DFD یا جریان داده می باشد. DFD نمایش گرافیکی از جریان داده ها درون یک سیستم است.**
- این نمودار نشان می‌دهد که چگونه داده‌های ورودی از طریق فرآیندهای مختلف به خروجی تبدیل می‌شوند و منابع داده، مخازن ذخیره‌سازی داده و مقاصد داده در سیستم کدام‌اند. DFD ها به‌طور مؤثری داده‌های ورودی، خروجی و داده‌های ذخیره‌شده در یک سیستم را به تصویر می‌کشند.

68

اجزای اصلی ترسیم DFD



موجودیت خارجی : با مستطیل نشان داده میشود. مانند مشتری، مدیر سیستم

فرایند: که با یک دایره نشان داده می شود. مانند فرایند رزرو بلیط.

مخازن داده: با یک مستطیل سه ضلعی یا دو خط موازی نشان داده میشود. جایی که داده ها ذخیره میشوند را نشان میدهد

جریان داده: با یک پیکان نشان داده میشود. که جریان انتقال داده را نشان میدهد.

69

سطوح در نمودار های جریان داده (DFD)

1. DFD را می‌توان به سطوح مختلفی تقسیم کرد که هر کدام درجه‌ای متفاوت از جزئیات را درباره سیستم ارائه می‌دهند.

   DFD سطح صفر

   DFD سطح ۱

   DFD سطح ۲

   DFD سطح ۳

2. انتخاب سطح مناسب DFD به میزان پیچیدگی سیستم و سطح جزئیاتی که برای درک آن لازم است بستگی دارد.
3. سطوح بالاتر DFD مثل سطح صفر، دید کلی و اجمالی از سیستم را ارائه می‌دهند، در حالی که سطوح پایین‌تر (مثل سطح ۲ یا ۳) اطلاعات دقیق‌تری از فرآیندها، جریان‌های داده و مخازن داده ارائه می‌کنند.
4. ترکیب چند سطح مختلف از DFD می‌تواند درک کاملی از سیستم به تحلیلگر بدهد.

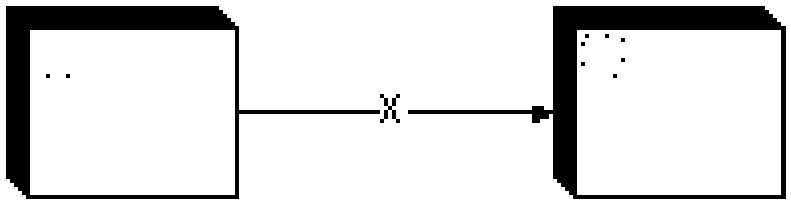
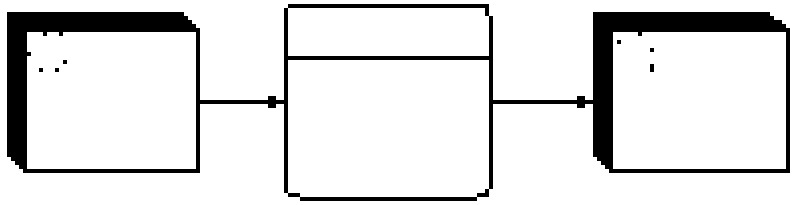
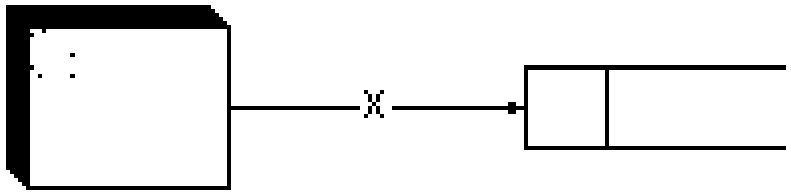
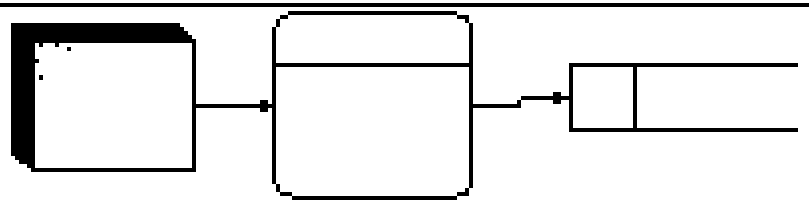
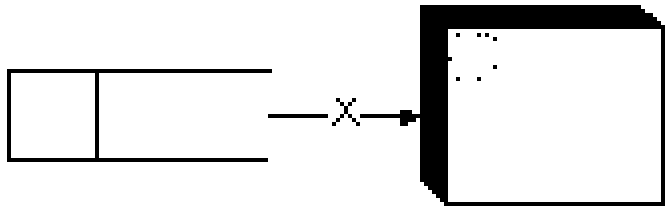
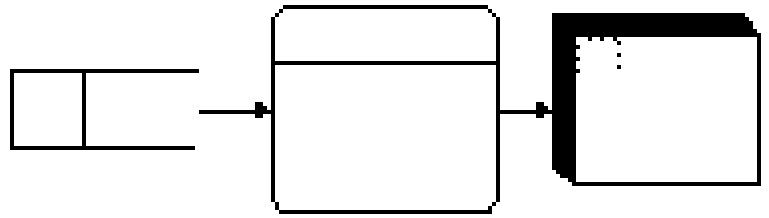
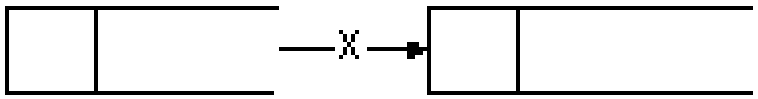
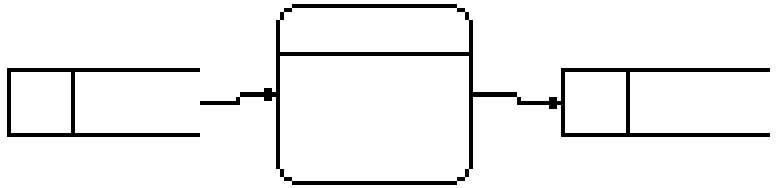
قوانین و اصول رسم نمودار جریان داده

1. **هر جریان داده باید به/از یک فرآیند برود :** یعنی نمی‌توانیم بین دو موجودیت یا دو مخزن داده پیکان رسم کنیم و حتما باید بین آنها حداقل یک فرایند باشد.
2. **مخازن داده باید فقط با فرایند ارتباط داشته باشد:** یعنی بین دو مخزن داده نمیتواند پیکان رسم شود.
3. **هر فرایند باید ورودی و خروجی داشته باشد:** یعنی نباید فقط داده بگیرد یا داده بدهد بلکه باید حداقل یکی از هر دو راداشته باشد. یعنی یک پیکان ورودی و یک پیکان خروجی حداقل داشته باشد.
4. **سطوح مختلف DFD باید سازگار باشند:** مثلاً درDFD سطح صفر اگر فرآیندی داریم به نام "پردازش سفارش"، در سطح ۱ باید همان کار را با جزئیات بیشتری نشان بدهیم . یا مثلا اگر در DFD سطح صفر یک موجودی داریم به نام مشتری باید در DFD سطح یک هم همین موجودی را داشته باشیم.
5. **شماره‌گذاری فرآیندها در سطوح مختلف DFD باید رعایت شود:**
   - DFD سطح ۰: فرآیندها مثل ۱، ۲، ۳
   - DFD سطح ۱: فرآیندهای زیرمجموعه مثلاً ۱.۱، ۱.۲
6. **نام‌ها باید واضح، قابل فهم و بدون ابهام باشند:**
   - نام فرآیند: فعل + اسم ← مثل "بررسی درخواست" یا "پردازش سفارش"
   - نام جریان داده: مثل "فرم ثبت‌نام"، "اطلاعات کاربر"

71

اشتباهات رایج در رسم نمودار DFD

Table 2: Common data flow diagramming mistakes

اشتباه	صحیح	توضیحات
		دو موجودیت خارجی نباید بهم وصل شوند
		موجودیت خارجی و مخزن داده ها نباید بهم وصل شوند.
		
		دو مخزن داده نباید بهم وصل شوند

DFD سطح صفر (Level 0 DFD)

- سطح صفر، بالاترین سطح از نمودار جریان داده است که نمایی کلی از کل سیستم ارائه می‌دهد. در این سطح، فرآیندهای اصلی، جریان‌های داده و مخازن داده بدون ورود به جزئیات داخلی آن‌ها نمایش داده می‌شوند. در سطح صفر مخازن داده نمایش داده نمیشوند.

- این نوع DFD با عنوان نمودار زمینه Context Diagram نیز شناخته می‌شود.

- در DFD سطح صفر، کل سیستم به‌صورت یک دایره واحد نمایش داده می‌شود و جریان داده‌ها بین موجودیت‌های خارجی و سیستم با فلش‌ها مشخص می‌گردد.

- تمرکز این سطح بر تعاملات سطح بالا با محیط بیرونی است، نه جزئیات درونی فرآیندها.

73 73

DFD سطح یک (DFD Level 1)

- نمودار جریان داده سطح یک، نمای دقیق‌تری از سیستم ارائه می‌دهد.
- در این سطح، فرآیند اصلی که درDFD سطح صفر شناسایی شده بود، به زیرفرآیندهای کوچک‌تر تقسیم می‌شود.
- هر زیرفرآیند به‌صورت یک فرآیند جداگانه در DFD سطح یک نمایش داده می‌شود.
- همچنین جریان‌های داده و مخازن داده مربوط به هر زیرفرآیند نیز در این سطح نشان داده می‌شود.
- در نمودار سطح یک، نمودار سطح یک به چندین دایره یا فرآیند تقسیم می‌شود. این سطح، عملکردهای اصلی سیستم را برجسته می‌کند و فرآیند کلی سطح صفر را به اجزای کاربردی قابل فهم تبدیل می‌نماید.
- برای مثال، اگر DFD سطح صفر نشان‌دهنده یک سیستم پرداخت باشد، سطح یک ممکن است آن را به زیرفرآیندهایی مانند: پردازش پرداخت، تولید فاکتور، و تأیید پرداخت تقسیم کند.

74

DFD سطح دو (DFD Level 2)

- نمودار جریان داده سطح دو، نمایی بسیار دقیق‌تر از سیستم ارائه می‌دهد. در این سطح، زیرفرآیندهایی که در DFD سطح یک شناسایی شده‌اند، به **زیرفرآیندهای جزئی‌تر** تجزیه می‌شوند.
- هر زیرفرآیند در سطح یک، در DFD سطح دو به صورت یک فرآیند مجزا نمایش داده می‌شود و **جریان‌های داده و مخازن داده** مرتبط با آن نیز در این سطح مشخص می‌گردند.
- استفاده از DFD سطح دو زمانی رایج است که سیستم **پیچیده** باشد و نیاز به **تجزیه دقیق‌تر عملکردها** وجود داشته باشد.
- این سطح اطلاعات جزئی‌تر و فنی‌تری در مورد کارکرد سیستم ارائه می‌دهد و برای مستندسازی نیازمندی‌ها و تهیه مستندات فنی بسیار مناسب است.
- برای مثال، در یک سیستم پردازش پرداخت، DFD سطح دو ممکن است فرآیند «پردازش پرداخت» را به مراحل مشخص‌تری مانند: **تأیید پرداخت، برداشت وجه، و تکمیل تراکنش** تجزیه کند.

DFD سطح سه (DFD Level 3)

- نمودار جریان داده سطح سه، **جزئی‌ترین سطح DFD** است که نمایی دقیق از فرآیندها، جریان‌های داده و مخازن داده در سیستم ارائه می‌دهد.
- این سطح معمولاً برای **سیستم‌های بسیار پیچیده** به کار می‌رود، جایی که نیاز به درک عمیق و جزئی از عملکرد سیستم وجود دارد.
- در DFD سطح سه، هر فرآیند با توصیف دقیقی از **داده‌های ورودی، نحوه پردازش، و خروجی‌ها** نمایش داده می‌شود. همچنین جریان‌های داده و مخازن داده مرتبط با هر فرآیند نیز به‌طور کامل نشان داده می‌شوند، و تصویری جامع از عملکرد داخلی سیستم را ارائه می‌کنند.
- این سطح برای **تحلیل دقیق فرآیندها**، توسعه سیستم در سطح فنی، و تهیه **مستندات مهندسی دقیق** بسیار مفید است.

76

مزایای استفاده از نمودار های جریان داده (DFD)

- **درک آسان:** نمودارهای جریان داده نمایش‌های گرافیکی هستند که درک آن‌ها ساده است و به‌راحتی می‌توان آن‌ها را برای افراد غیرفنی یا ذی‌نفعان توضیح داد.
- **بهبود تحلیل سیستم:** DFDها در تحلیل فرآیندها و جریان داده‌های سیستم مفیدند. آن‌ها می‌توانند در شناسایی ناکارآمدی‌ها، تکرارها و سایر مشکلات موجود در سیستم کمک کنند.
- **پشتیبانی از طراحی سیستم:** DFDها برای طراحی ساختار و معماری سیستم کاربرد دارند و کمک می‌کنند تا طراحی سیستم مطابق با نیازهای کاربران انجام شود.
- **امکان تست و ارزیابی:** DFDها ورودی‌ها و خروجی‌های سیستم را مشخص می‌کنند که این موضوع به تست و ارزیابی عملکرد سیستم کمک می‌کند.
- **سهولت در مستندسازی:** DFDها نمایش تصویری از سیستم ارائه می‌دهند و این موضوع باعث می‌شود مستندسازی و نگهداری سیستم در طول زمان ساده‌تر شود.

77

معایب استفاده از نمودار جریان داده (DFD)

- **زمان‌بر بودن:**
  تهیه نمودارهای جریان داده مخصوصاً برای سیستم‌های پیچیده می‌تواند فرآیندی زمان‌بر باشد.
- **تمرکز محدود:**
  DFD ها عمدتاً روی جریان داده‌ها تمرکز دارند و ممکن است جنبه‌های مهم دیگر سیستم مانند طراحی رابط کاربری، امنیت سیستم یا عملکرد سیستم را پوشش ندهند.
- **نگهداری دشوار:**
  با تغییر و تکامل سیستم، به‌روزرسانی DFD ها ممکن است دشوار و زمان‌بر شود و نمودارها به سرعت قدیمی شوند.
- **نیاز به تخصص فنی:**
  هرچند فهم DFD ها ساده است، اما رسم دقیق و صحیح آن‌ها نیازمند دانش فنی و آشنایی با سیستم مورد تحلیل است.

78

ساختار شکست کار
WBS(Work Breakdown Structure)

(WBS) به معنای "ساختار شکست کار "است. WBS پروژه را به فعالیت‌ها و وظایف کوچکتر تقسیم می‌کند.

اهمیت WBS

1. **ایجاد وضوح و شفافیت.**
2. **تسهیل تخصیص منابع.**
3. **امکان پیگیری و کنترل پیشرفت پروژه.**
4. **کاهش احتمال افزایش غیرمجاز محدوده پروژه.**

79

برای تهیه WBS چطور پروژه را به مراحل و فعالیت های کوچکتر تقسیم کنیم؟

تقسیم پروژه به مراحل و فازهای اصلی: تقسیم پروژه به مراحل کلی مانند تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی، تست و استقرار.

تقسیم هر مرحله به فعالیت‌های کوچکتر: تعریف فعالیت‌های مشخص در هر مرحله.

ایجاد سلسله مراتب (Hierarchy): ایجاد ساختار سلسله مراتبی از فعالیت‌ها.

80

الزامات شکست کار:

قواعد اصلی ساختار شکست کار(WBS)

- **قاعده ی ۱۰۰ درصد رعایت شده باشد**
- **مبتنی بر تحویل دادنی ها باشد.**

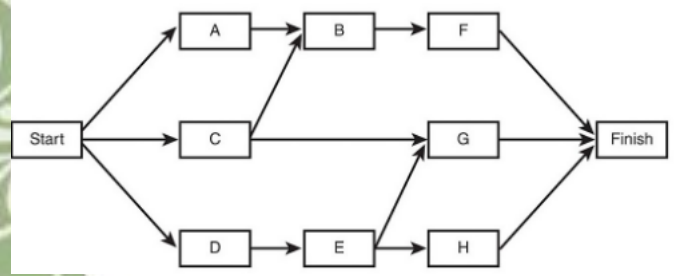
قاعده ۱۰۰درصد: یعنی هر مجموعه عنصر در ساختار شکست کاریا فعالیت های زیر مجموعه ی آن باید دقیقا صددرصد کارهای عنصر مادر خود را شامل شوند. نه کمتر و نه بیشتر

یعنی اگر عنوان یک فعالیت اصلی را به زیر فعالیتهای کوچکتر شکستیم، انجام همه ی آن فعالیتهای کوچکتر باید منجر به انجام فعالیت اصلی شود.

نمودار شبکه‌ای در کنترل پروژه

نمودارهای شبکه‌ای پروژه (Project Network Diagrams): کلیت پروژه را با استفاده از المان‌های گرافیکی نمایش می‌دهند .این نمودارها، از چندین دایره و پیکان متصل به هم تشکیل می‌شوند.

به هر دایره در نمودارهای شبکه‌ای، یک گره می‌گویند .گره‌ها، بیانگر فعالیت‌های یک پروژه و پیکان‌ها، نشان‌دهنده رابطه بین فعالیت‌ها هستند.



یکی از متداول‌ترین روشهای رسم نمودارهای شبکه‌ای نمودار PDM می باشد.

(Precedence Diagramming Method | PDM)

نمودارهای شبکه‌ای در بسیاری از تکنیک‌های گرافیکی مدیریت و کنترل پروژه نظیر تکنیک بررسی و ارزیابی برنامه (PERT) و روش مسیربحرانی (CPM) کاربرد دارند.

82

مسیر بحرانی CPM (Critical Path Method)

طولانی ترین زمان اجرای فعالیت های یک پروژه را مسیر بحرانی میگویند.

برای محاسبه ی مسیر بحرانی نیاز به زمان هر فعالیت داریم.

اگر زمان هریک از فعالیت های مسیر بحرانی زیاد یا کم بشود در زمان نهایی پروژه تاثیر می گذارد.

یک پروژه میتواند چند مسیربحرانی داشته باشد.

برای محاسبه ی مسیر بحرانی ، کلیه مسیر های نمودار PDM را محاسبه میکنیم سپس طولانی ترین مسیر، مسیر بحرانی نامیده میشود.

روش بررسی و ارزیابی برنامه PERT

Program Evaluation Review Technique

باتوجه به اینکه براورد زمان انجام فعالیت های یک پروژه دقیق نیست بهتر است برای محاسبه ی زمان نهایی پروژه از زمانهای تخمینی خوشبینانه و بدبینانه و زمان واقعی استفاده کنیم و زمان تقریبی را تخمین بزنیم.

زمان واقع بینانه زمانی است که از روشهای استاندارد بدست می آوریم.

تکنیک PERT: استفاده از سه تخمین (خوش‌بینانه، بدبینانه،واقع بینانه) برای محاسبه زمان تقریبی یک پروژه است.

زمان تقریبی یا محتمل = (خوش‌بینانه + (۴ × واقع‌بینانه) + بدبینانه) ÷ ۶

84

گانت چارت(Gantt chart)

گانت چارت یک نمودار زمانی است که نشان میدهد:

- هر فعالیت **چه زمانی شروع میشود.**
- **چه زمانی پایان می یابد.**

اجزای نمودار:

- محور **افقی: زمان**
- محور عمودی: **فعالیت ها**
- نوار افقی: **مدت انجام هر فعالیت**

گانت چه چیزی را نشان میدهد:

- برنامه ریزی پروژه
- همپوشانی فعالیت ها
- وضعیت کلی پروژه برای مدیران

چه چیزی را نشان نمیدهد:

- **محاسبه مسیربحرانی**
- **تحلیل وابستگیها**

85

ارتباط گانت چارت با
PERT, CPM, PDM, WBS

- فعالیت های گانت از WBS
- ترتیب اجرا از PDM
- زمان فعالیت ها از CPM یا PERT

گانت چارت ابزار نمایش و گزارش است و ابزار تحلیل نیست

نکته: تفاوت گانت چارت و ساختار شکست کار یا همان WBS اینست که در گانت چارت زمان هر فعالیت را داریم

86

مثال از گانت چارت

فازبندی	درصد	شرح خدمت	از تاریخ	تا تاریخ	روز	ماه 1				ماه 2				ماه 3				ماه 4				ماه 5			
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
فاز ۱	10	تحلیل نیازمندیها و جلسات تکمیلی در صورت نیاز			0																				
		تهیه پروتوتایپ فرم ها																							
		ایجاد بانک های اطلاعاتی و دیتامدل			0																				
		ایجاد جداول پایه			0																				
فاز ۲	20	پیاده سازی فرمهاهای پایه شامل:			0																				
		فرم تعریف حسابها			0																				
		فرم تعریف اشخاص			0																				
		فرم تعریف دست چک			0																				
		فرم تعریف انواع چک			0																				
		فرم اعطا و اخذ مجوز های دسترسی			0																				
		سایر فرم های پایه																							
فاز ۳	20	پیاده سازی فرم ثبت چک			0																				
		پیاده سازی فرم انواع واریزی ها			0																				
		پیاده سازی import(خواندن از اکسل)			0																				
		پیاده سازی Export(ارسال به سامانه ی مالی)			0																				
فاز ۴	30	پیاده سازی گزارشات شامل			0																				
		انواع گزارشات چک و واریزی ها																							
		انواع گزارشات فصلی																							
		انواع گزارشات import																							
		انواع گزارشات export																							
		گزارش لاگ کاربران																							
فاز ۵	20	خاتمه پروژه شامل																							
		استقرار در سرور تست																							
		کانورت اطلاعات																							
		کنترل کیفیت اولیه																							
		استقرار در سرور اصلی																							
		کنترل کنیفیت نهایی																							
		مستند آموزش راهبری																							
		مستند آموزش کاربری																							
		مستند نصب و راه اندازی																							

مثال از PERT, CPM, PDM, WBS

مثال:

۱- wbs را برای پروژه ی توسعه ی وب سایت فروشگاه را بنویسید.
۲- فعالیت ها را کد گذاری کنید.
۳- مدت زمان هر فعالیت را با توجه به استانداردها و ظرفیت تیم برآورد کنید.
۴- وابستگی های فعالیت ها را تعیین کنید.
۵- نمودار PDM را ترسیم کنید.
۶- مسیر بحرانی یا CPM را ترسیم کنید.
۷- با فرض اینکه زمان فعالیت های پروژه خوشبینانه و بدبینانه داشته باشندبا استفاده از روش pert مجدد زمان محتمل هر فعالیت را محاسبه کنید.

88

پاسخ مثال:

۱- WBS (ستون A)

A
فعالیت ها
۱. شروع و برنامه‌ریزی پروژه
۱.۱ سند هدف و دامنه پروژه
۱.۲ سند برنامه زمان‌بندی پروژه
۲. تحلیل نیازمندی‌ها
۲.۱ قابلیت‌های سیستم
۲.۱.۱ مشخصات قابلیت ثبت‌نام و ورود کاربران
۲.۱.۲ مشخصات قابلیت مدیریت پروژه‌ها
۲.۱.۳ مشخصات قابلیت مدیریت وظایف
۲.۲ مستندات نیازمندی‌ها
۲.۲.۱ سند نیازمندی‌های سیستم (SRS)
۳. طراحی سیستم
۳.۱ مدل پایگاه داده سیستم (ERD)
۳.۲ طراحی رابط کاربری
۴. پیاده‌سازی
۴.۱ پیاده‌سازی Backend
۴.۱.۱ ماژول مدیریت کاربران
۴.۱.۲ ماژول مدیریت پروژه‌ها
۴.۱.۳ ماژول مدیریت وظایف
۴.۲ پیاده‌سازی Frontend
۴.۲.۱ رابط کاربری کاربران
۴.۲.۲ رابط کاربری پنل مدیریت
۵. تست و تحویل
۵.۱ گزارش نتایج تست سیستم
۵.۲ نسخه نهایی سامانه

پاسخ مثال:

۱- WBS (ستون A)
۲- کد گذاری فعالیت ها (ستون B)

B	A
	فعالیت ها
	۱. شروع و برنامه‌ریزی پروژه
a	۱.۱ سند هدف و دامنه پروژه
b	۱.۲ سند برنامه زمان‌بندی پروژه
	۲. تحلیل نیازمندی‌ها
	۲.۱ قابلیت‌های سیستم
c	۲.۱.۱ مشخصات قابلیت ثبت‌نام و ورود کاربران
d	۲.۱.۲ مشخصات قابلیت مدیریت پروژه‌ها
e	۲.۱.۳ مشخصات قابلیت مدیریت وظایف
	۲.۲ مستندات نیازمندی‌ها
f	۲.۲.۱ سند نیازمندی‌های سیستم (SRS)
	۳. طراحی سیستم
g	۳.۱ مدل پایگاه داده سیستم (ERD)
h	۳.۲ طراحی رابط کاربری
	۴. پیاده‌سازی
	۴.۱ پیاده‌سازی Backend
i	۴.۱.۱ ماژول مدیریت کاربران
j	۴.۱.۲ ماژول مدیریت پروژه‌ها
k	۴.۱.۳ ماژول مدیریت وظایف
	۴.۲ پیاده‌سازی Frontend
l	۴.۲.۱ رابط کاربری کاربران
m	۴.۲.۲ رابط کاربری پنل مدیریت
	۵. تست و تحویل
n	۵.۱ گزارش نتایج تست سیستم
o	۵.۲ نسخه نهایی سامانه

90

پاسخ مثال:

۱- WBS (ستون A)
۲- کد گذاری فعالیت ها (ستون B)
۳- مدت زمان هر فعالیت(ستونC)

C	B	A
مدت روز		**فعالیت ها**
		۱. شروع و برنامه‌ریزی پروژه
۲	a	۱.۱ سند هدف و دامنه پروژه
۲	b	۱.۲ سند برنامه زمان‌بندی پروژه
		۲. تحلیل نیازمندی‌ها
		۲.۱ قابلیت‌های سیستم
۳	c	۲.۱.۱ مشخصات قابلیت ثبت‌نام و ورود کاربران
۳	d	۲.۱.۲ مشخصات قابلیت مدیریت پروژه‌ها
۳	e	۲.۱.۳ مشخصات قابلیت مدیریت وظایف
		۲.۲ مستندات نیازمندی‌ها
۲	f	۲.۲.۱ سند نیازمندی‌های سیستم (SRS)
		۳. طراحی سیستم
۳	g	۳.۱ مدل پایگاه داده سیستم (ERD)
۴	h	۳.۲ طراحی رابط کاربری
		۴. پیاده‌سازی
		۴.۱ پیاده‌سازی Backend
۵	i	۴.۱.۱ ماژول مدیریت کاربران
۶	j	۴.۱.۲ ماژول مدیریت پروژه‌ها
۶	k	۴.۱.۳ ماژول مدیریت وظایف
		۴.۲ پیاده‌سازی Frontend
۵	l	۴.۲.۱ رابط کاربری کاربران
۵	m	۴.۲.۲ رابط کاربری پنل مدیریت
		۵. تست و تحویل
۴	n	۵.۱ گزارش نتایج تست سیستم
۱	o	۵.۲ نسخه نهایی سامانه

91

D	C	B	A
وابستگی ها	مدت روز		فعالیت ها
			۱. شروع و برنامه‌ریزی پروژه
a	۲	a	۱.۱ سند هدف و دامنه پروژه
a	۲	b	۱.۲ سند برنامه زمان‌بندی پروژه
			۲. تحلیل نیازمندی‌ها
			۲.۱ قابلیت‌های سیستم
a	۳	c	۲.۱.۱ مشخصات قابلیت ثبت‌نام و ورود کاربران
a	۳	d	۲.۱.۲ مشخصات قابلیت مدیریت پروژه‌ها
a	۳	e	۲.۱.۳ مشخصات قابلیت مدیریت وظایف
			۲.۲ مستندات نیازمندی‌ها
c,d,e	۲	f	۲.۲.۱ سند نیازمندی‌های سیستم (SRS)
			۳. طراحی سیستم
f	۳	g	۳.۱ مدل پایگاه داده سیستم (ERD)
f	۴	h	۳.۲ طراحی رابط کاربری
			۴. پیاده‌سازی
			۴.۱ پیاده‌سازی Backend
g	۵	i	۴.۱.۱ ماژول مدیریت کاربران
g	۶	j	۴.۱.۲ ماژول مدیریت پروژه‌ها
g	۶	k	۴.۱.۳ ماژول مدیریت وظایف
			۴.۲ پیاده‌سازی Frontend
h,i	۵	l	۴.۲.۱ رابط کاربری کاربران
h,j,k	۵	m	۴.۲.۲ رابط کاربری پنل مدیریت
			۵. تست و تحویل
I,j,k,l,m	۴	n	۵.۱ گزارش نتایج تست سیستم
n	۱	o	۵.۲ نسخه نهایی سامانه

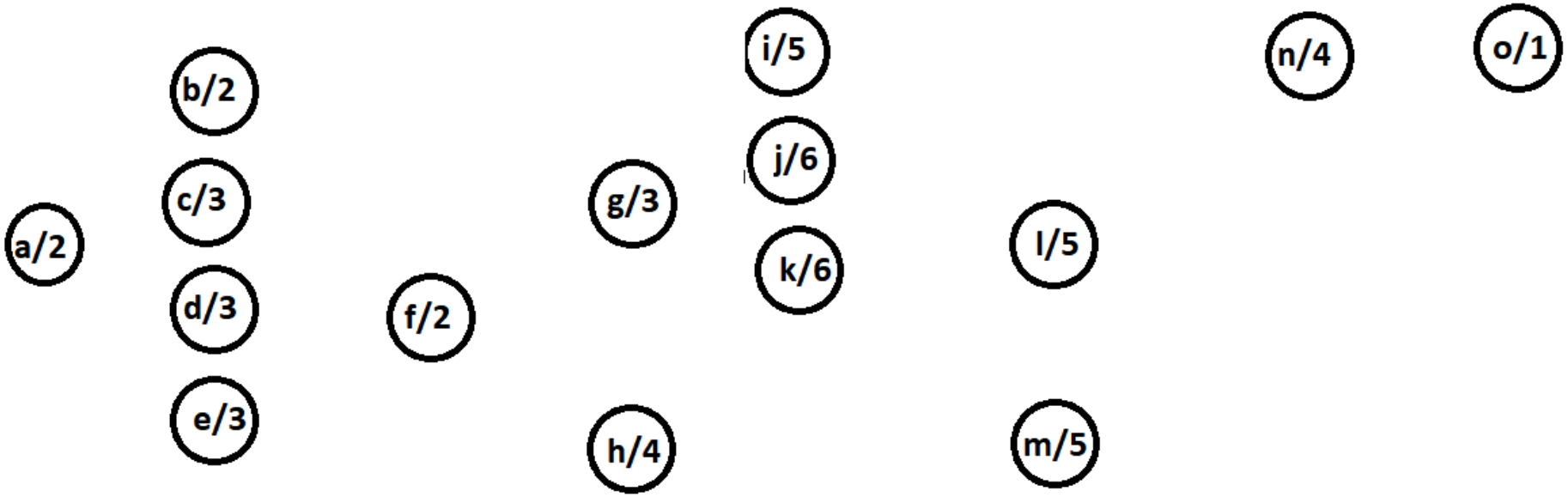
پاسخ مثال:

۱- WBS (ستون A)
۲- کد گذاری فعالیت ها (ستون B)
۳- مدت زمان هر فعالیت (ستونC)
۴- وابستگی فعالیت ها (ستونD)

92

پاسخ مثال:

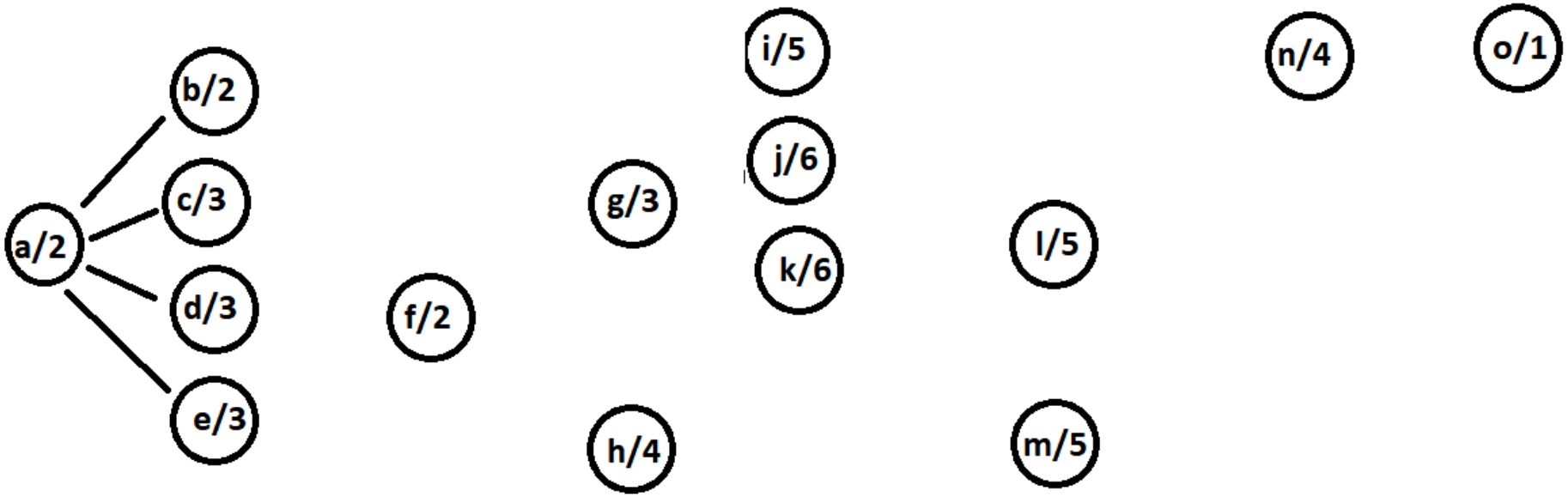
۵- مراحل نمودار PDM :



93

پاسخ مثال:

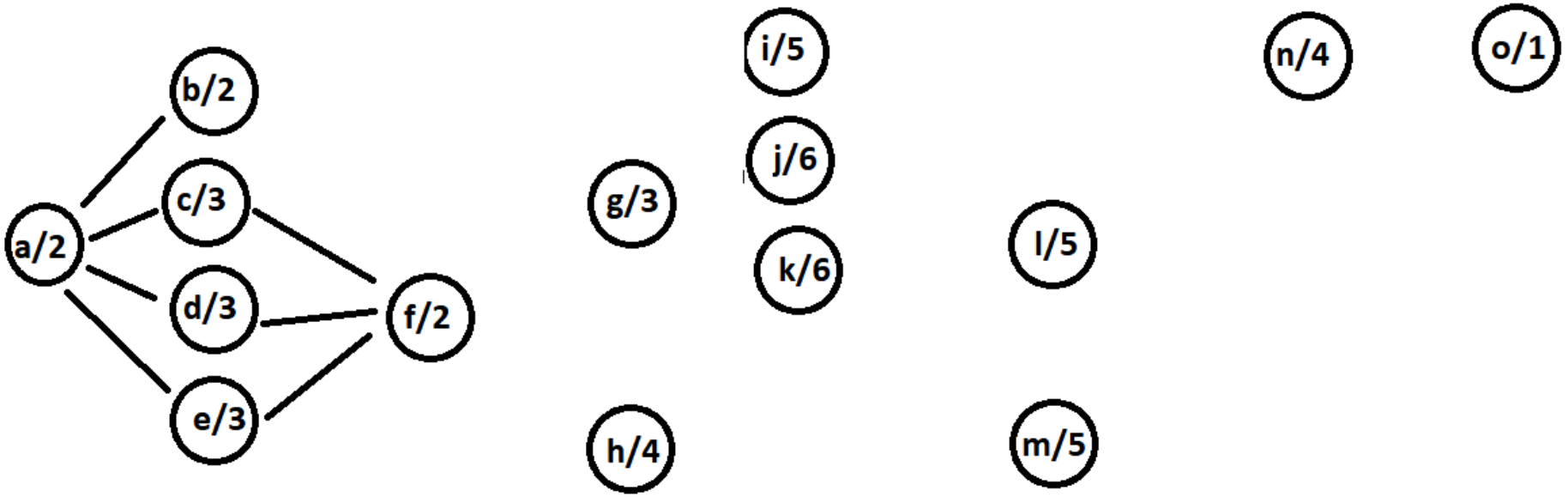
۵- مراحل نمودار PDM :



94

پاسخ مثال:

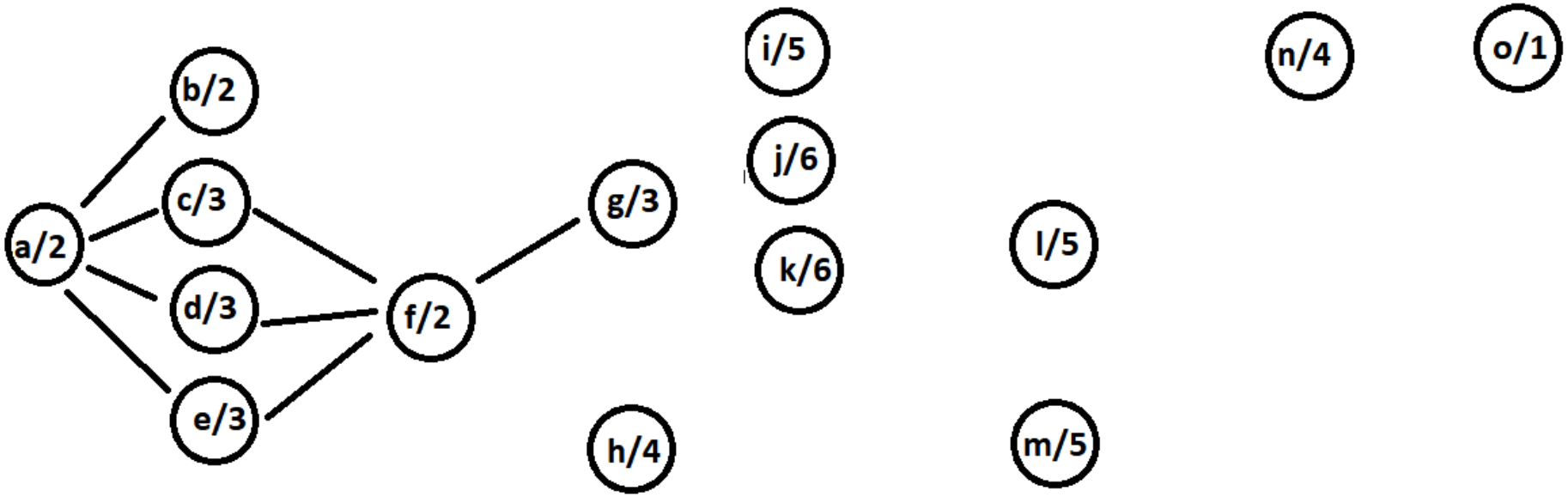
۵- مراحل نمودار PDM :



95

پاسخ مثال:

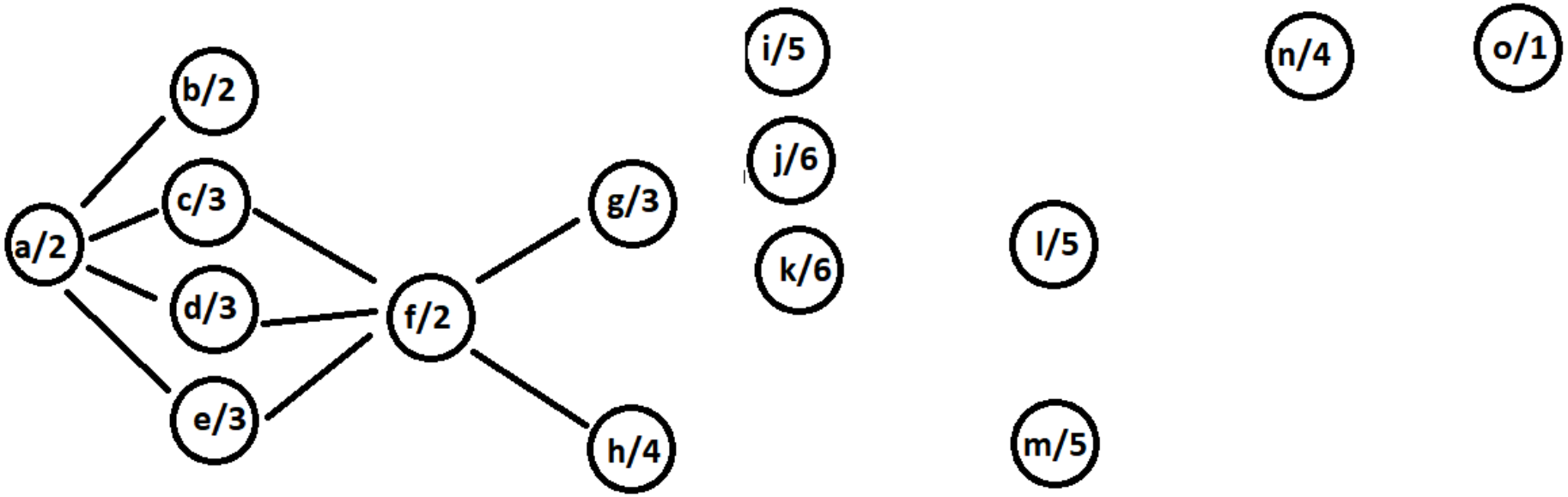
۵- مراحل نمودار PDM :



96

پاسخ مثال:

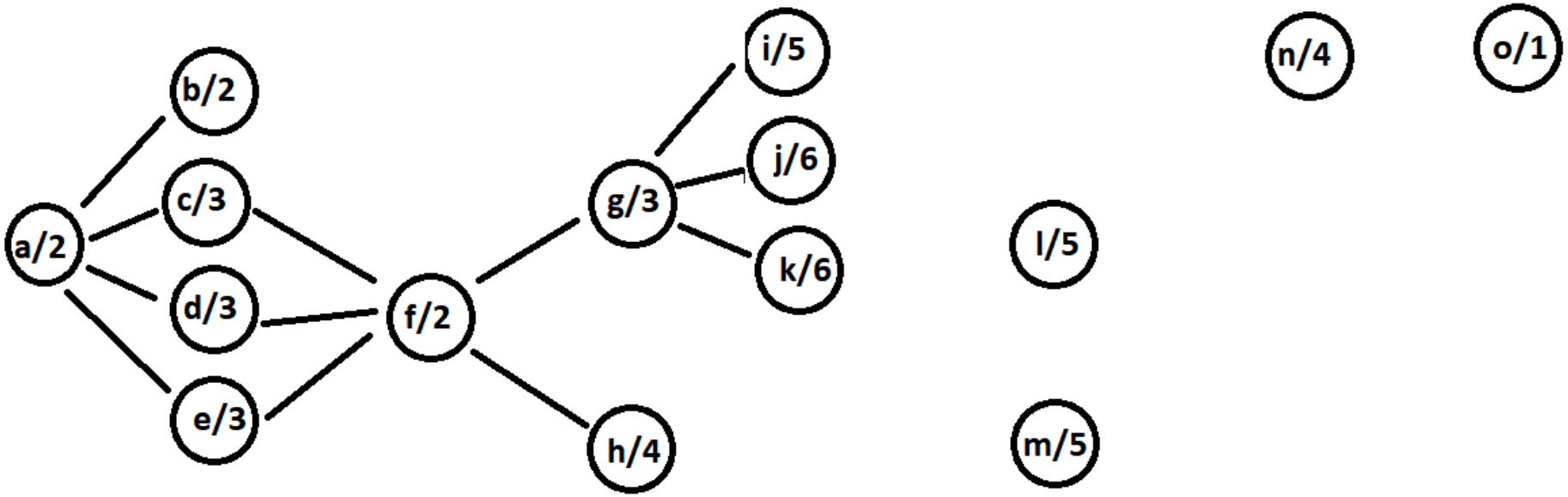
۵- مراحل نمودار PDM :



97

پاسخ مثال:

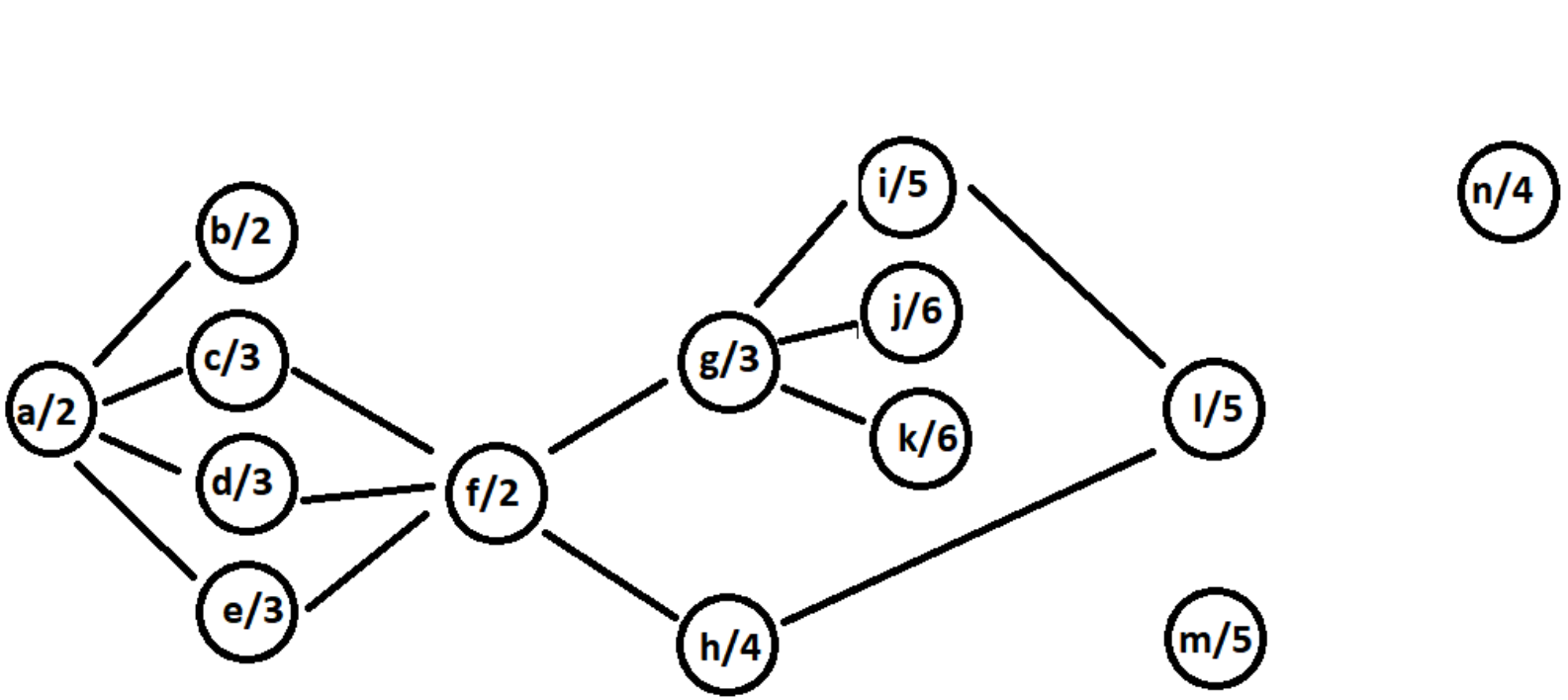
۵- مراحل نمودار PDM :



98

پاسخ مثال:

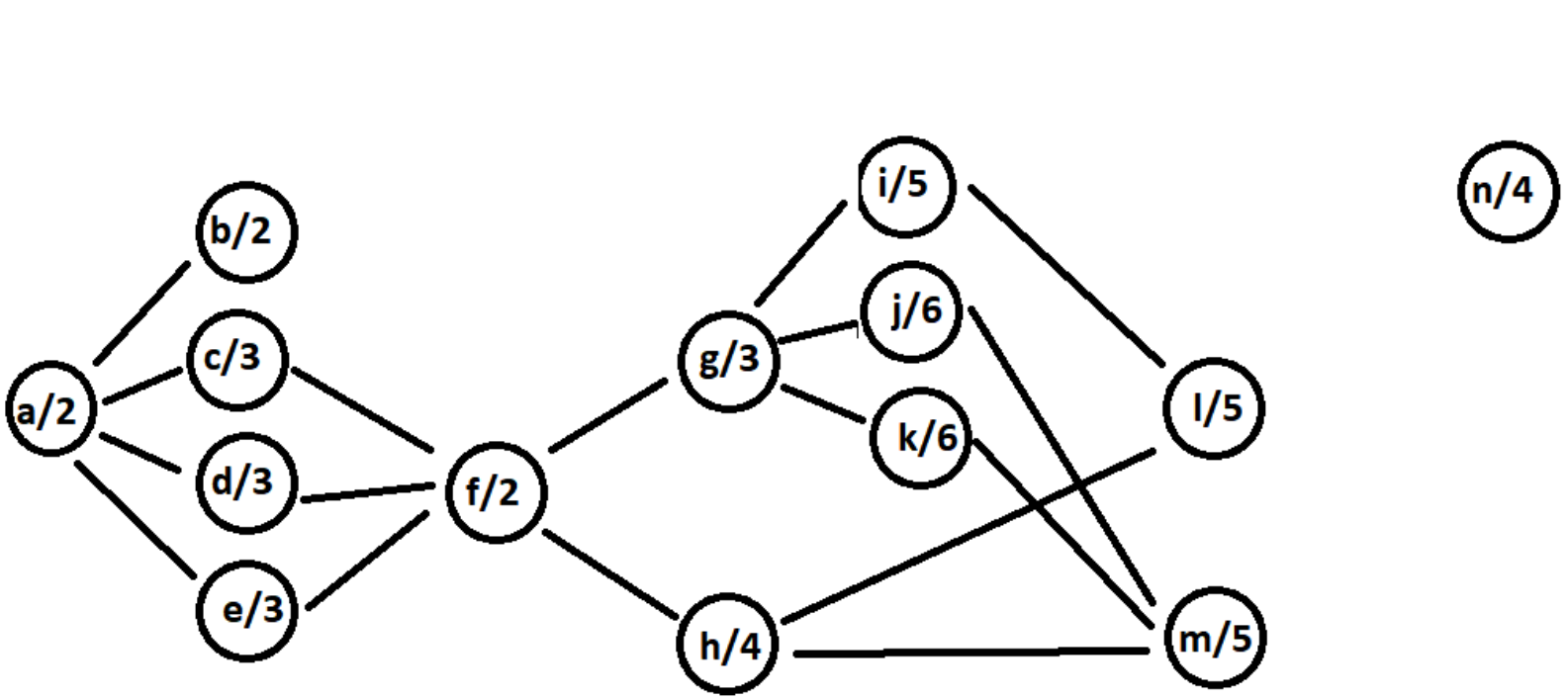
۵- مراحل نمودار PDM :



99

پاسخ مثال:

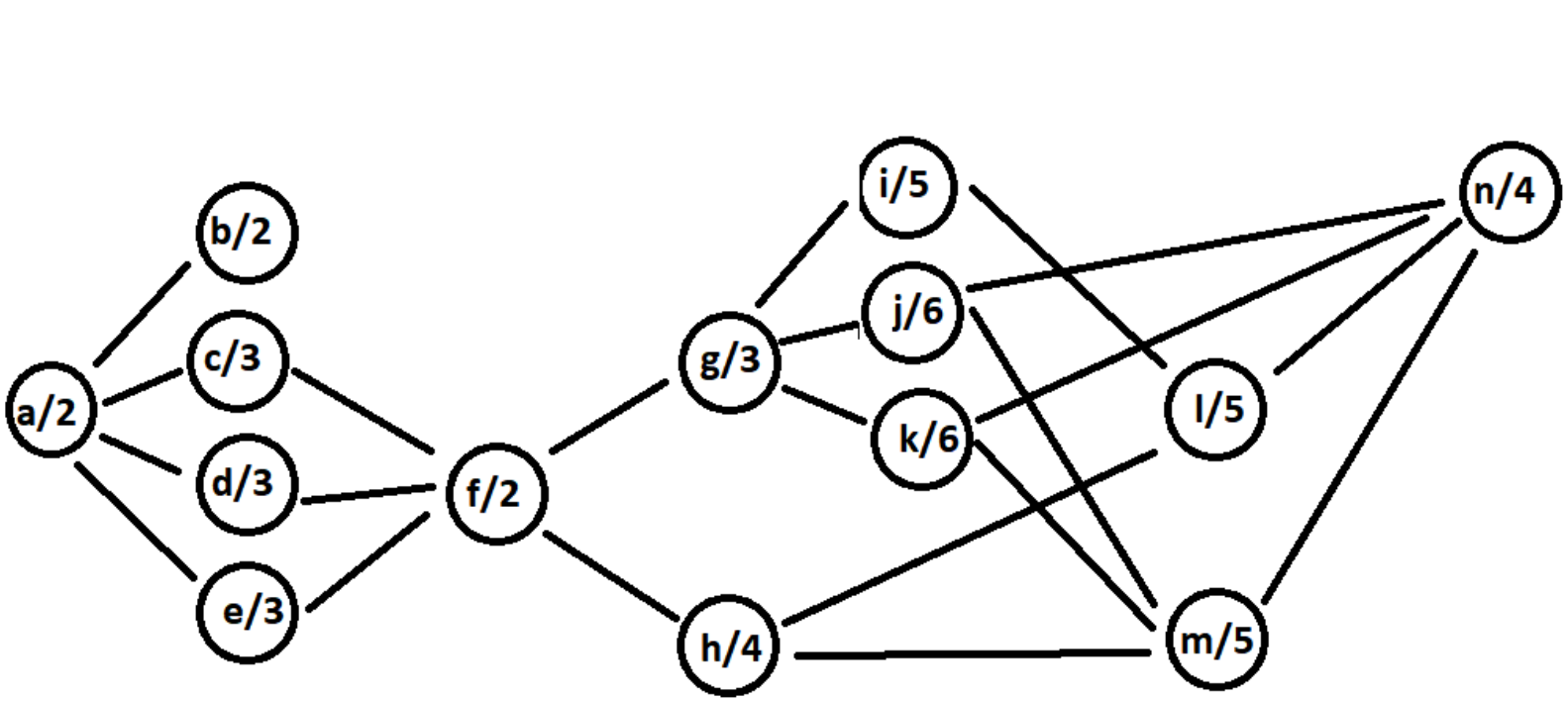
۵- مراحل نمودار PDM :



100

پاسخ مثال:

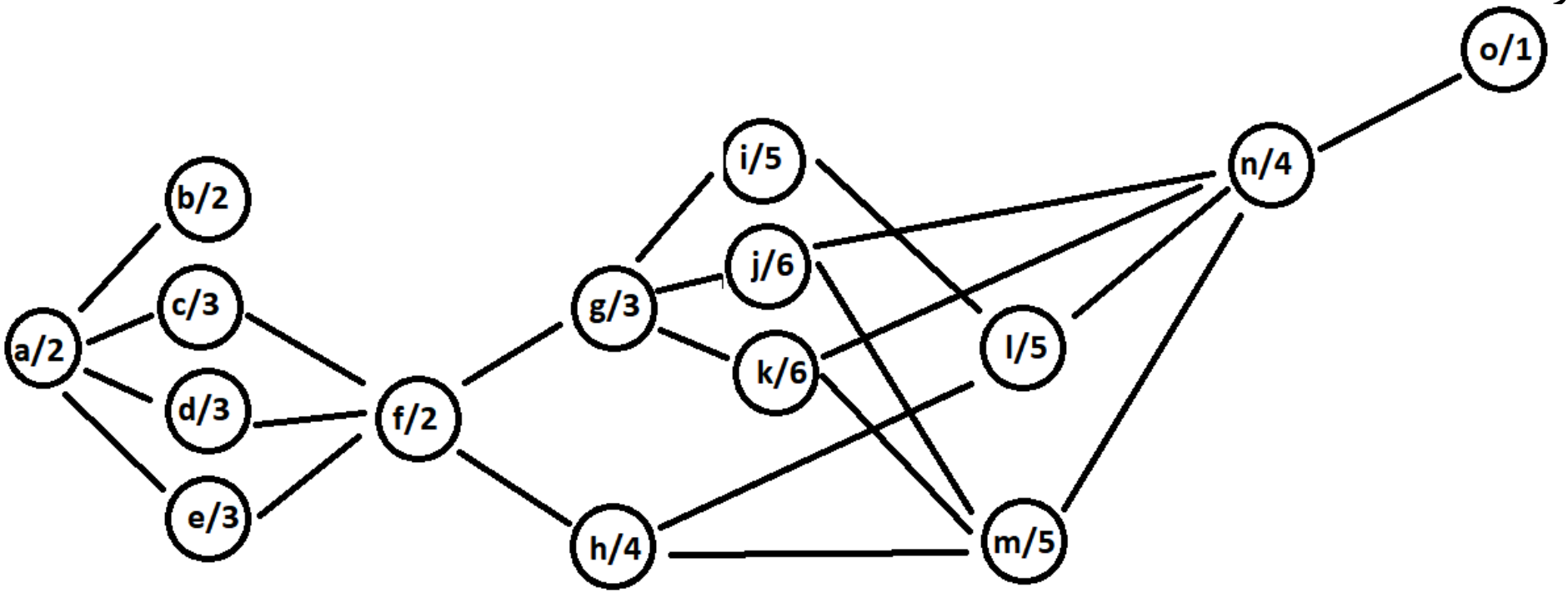
۵- مراحل نمودار PDM :



101

پاسخ مثال:

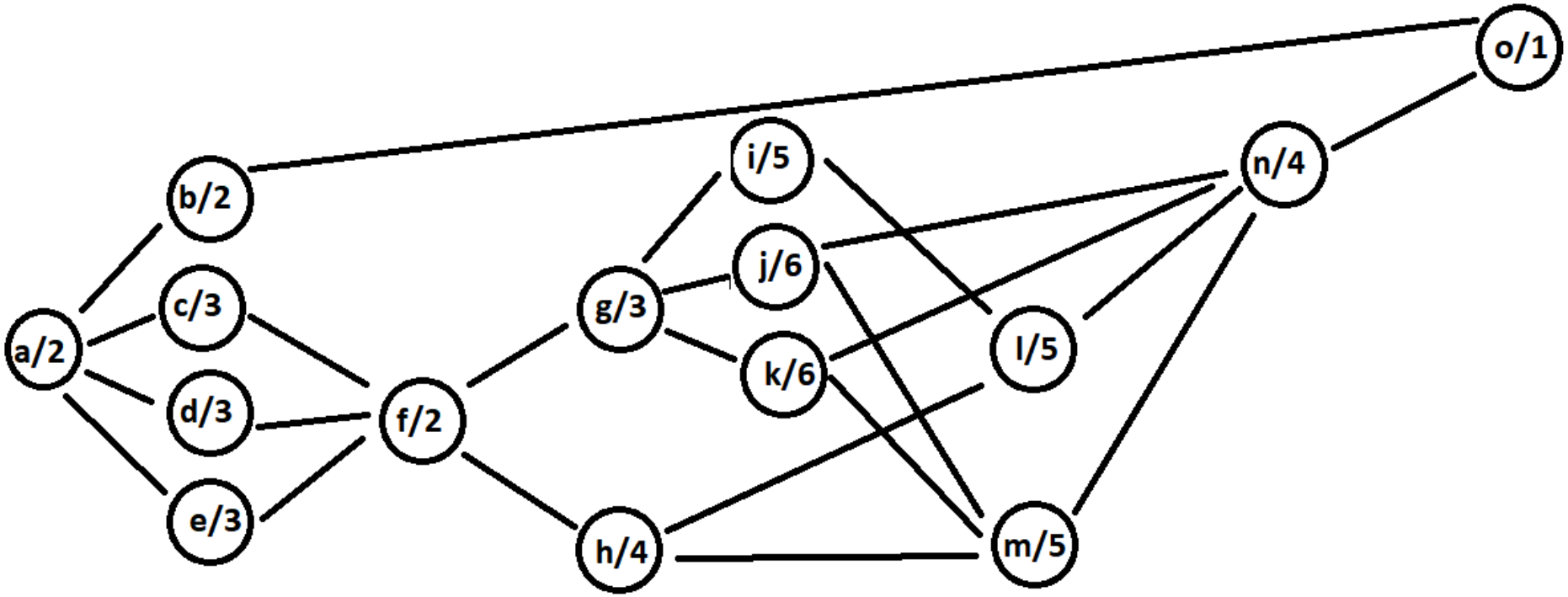
۵- مراحل نمودار PDM :



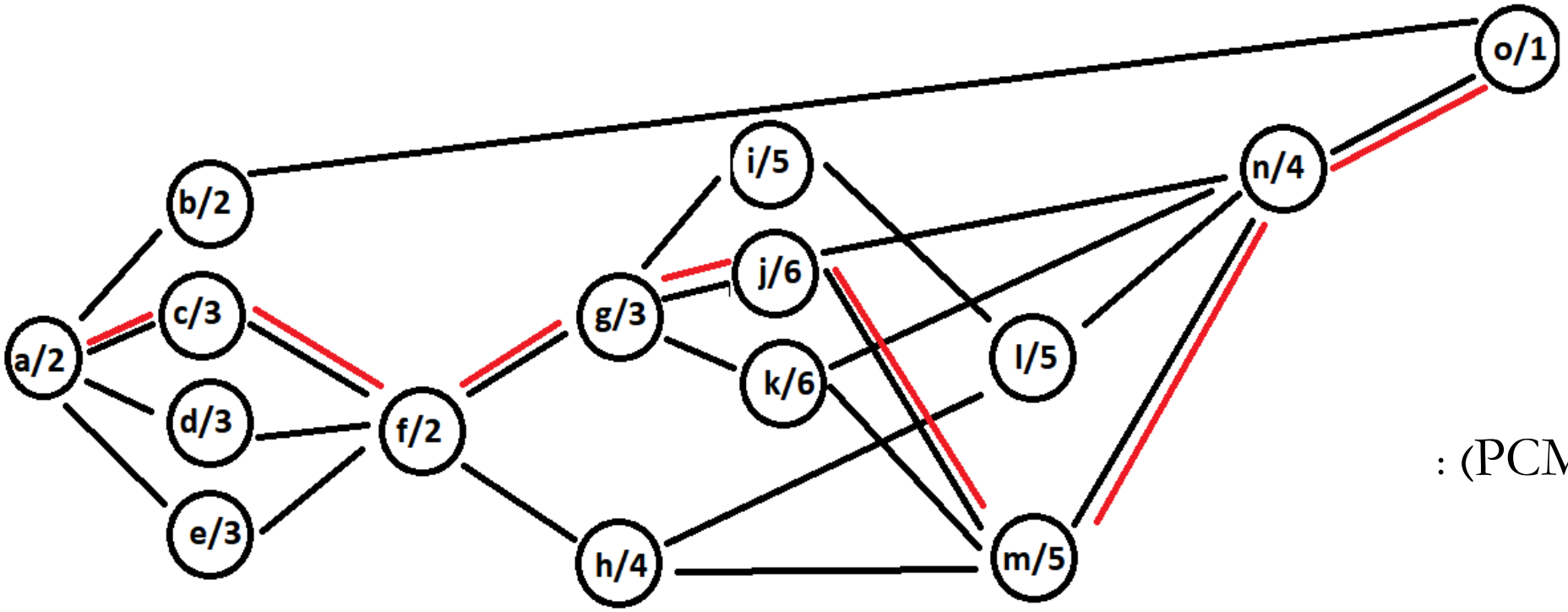
102

پاسخ مثال:

۵- مراحل نمودار PDM :



103



پاسخ مثال:

۶- مسیر بحرانی(PCM) :

یاداوری: در پاسخ به سوالات امتحانی تشریحی طول کلیه مسیرهای ممکن در نمودارPDM باید نوشته شود و طولانی ترین مسیر انتخاب شود.

مثال:

۷- محاسبه زمان محتمل هر فعالیت با استفاده از روش PERT(ستون G)

A	B	C	D	E	F	G
فعالیت ها		**مدت روز**	**وابستگی ها**	**خوشبینانه**	**بدبینانه**	**زمان محتمل**
۱. شروع و برنامه‌ریزی پروژه						
۱.۱ سند هدف و دامنه پروژه	a	۲	a	۲	۳	۲.۲
۱.۲ سند برنامه زمان‌بندی پروژه	b	۲	a	۲	۴	۲.۳
۲. تحلیل نیازمندی‌ها						۰
۲.۱ قابلیت‌های سیستم						۰
۲.۱.۱ مشخصات قابلیت ثبت‌نام و ورود کاربران	c	۳	a	۲	۴	۳.۰
۲.۱.۲ مشخصات قابلیت مدیریت پروژه‌ها	d	۳	a	۲	۴	۳.۰
۲.۱.۳ مشخصات قابلیت مدیریت وظایف	e	۳	a	۳	۵	۳.۳
۲.۲ مستندات نیازمندی‌ها						۰
۲.۲.۱ سند نیازمندی‌های سیستم (SRS)	f	۲	c,d,e	۲	۴	۲.۳
۳. طراحی سیستم						۰
۳.۱ مدل پایگاه داده سیستم (ERD)	g	۳	f	۲	۴	۳.۰
۳.۲ طراحی رابط کاربری	h	۴	f	۳	۶	۴.۲
۴. پیاده‌سازی						۰
۴.۱ پیاده‌سازی Backend						۰
۴.۱.۱ ماژول مدیریت کاربران	i	۵	g	۳	۷	۵.۰
۴.۱.۲ ماژول مدیریت پروژه‌ها	j	۶	g	۵	۸	۶.۲
۴.۱.۳ ماژول مدیریت وظایف	k	۶	g	۴	۷	۵.۸
۴.۲ پیاده‌سازی Frontend						۰
۴.۲.۱ رابط کاربری کاربران	l	۵	h,i	۴	۷	۵.۲
۴.۲.۲ رابط کاربری پنل مدیریت	m	۵	h,j,k	۳	۸	۵.۲
۵. تست و تحویل						۰
۵.۱ گزارش نتایج تست سیستم	n	۴	I,j,k,l,m	۳	۵	۴.۰
۵.۲ نسخه نهایی سامانه	o	۱	n	۱	۴	۱.۵

105