دانشگاه جامع امام حسین (ع)
دانشکده و پژوهشکده مهندسی فناوری اطلاعات و ارتباطات
گروه مهندسی کامپیوتر

عنوان درس:

# معماری نرم‌افزار

## Software Architecture (SA)

## فصل ۹: مستندسازی معماری نرم‌افزار – (Documenting Software Architectures)

**مدرس:**

**علی کریمی**

**(Ali Karimi)**

نیمسال: ۱-۱۴۰۰

a.karimi@ihu.ac.ir

# فهرست مطالب

- اهمیت مستندسازی معماری نرم‌افزار
- انتخاب دیدهای مرتبط
- مستندسازی دید
- مستندسازی بین دیدها
- استفاده از UML

# مقدمه

- در این فصل:
  - به منظور **مستندسازی معماری نرم‌افزار**، یک **قالب استاندارد** معرفی خواهد شد.
  - در این **قالب** مشخص می‌شود که **چه اطلاعاتی باید در مستند معماری** قرار داده شود.
  - **راهنمایی‌هایی در مورد چگونگی بدست آوردن اطلاعات موردنیاز** برای قرار گرفتن در **مستند معماری** ارائه خواهد شد.
  - نهایتاً، **نقش UML** در **مستندسازی معماری** بررسی خواهد شد.

# اهمیت مستندسازی معماری نرم‌افزار

- معماری نرم‌افزار نقش محوری در توسعه سیستم و سازماندهی فراورده‌های آن دارد.
- معماری نرم‌افزار، به عنوان یک طرح جامع و کلی برای پروژه و توسعه سیستم لحاظ می‌شود.
- نقش محوری معماری نرم‌افزار در توسعه سیستم سبب شده است که نحوهٔ تولید، درک و استفاده از معماری از اهمیت خاصی برخوردار شود.
- مستندسازی معماری نشان می‌دهد که:
  - ۱- نقشه توسعه سیستم چگونه است،
  - ۲- نحوهٔ انجام کار توسط تیم‌های مختلف (طراحان، پیاده‌سازان) چگونه است،
  - ۳. خصوصیات کیفی چگونه برآورده می‌شوند (با استفاده از چه تاکتیک‌هایی و چه الگوهایی؟)،
  - ۴- چگونه سیستم باید نگهداری و پشتیبانی شود. (به واسطه نگهداری ساختارهای کلیدی توسط معماری).

# اهمیت مستندسازی معماری نرم‌افزار (ادامه)

- معماری، یک فرآورده برای تحلیل اولیه است تا:
  - تضمین شود روش طراحی منجر به سیستم قابل قبول خواهد شد.
- مستندسازی معماری، یک مرحله اصلی و اساسی در شکل‌گیری معماری است.
- در صورتی‌که کسی نتواند معماری (هر چند کامل) را بفهمد،
  - معماری غیرقابل استفاده خواهد بود.
- در صورتی‌که می‌خواهید معماری قوی داشته باشید، آن را بدون ابهام، با جزئیات کافی و سازمان‌دهی‌شده مستندسازی نمائید به طوریکه دیگران آن را بفهمند.
- ایجاد معماری بدون خصوصیات فوق،
  - باعث بی‌نتیجه شدن تلاش‌ها خواهد شد (زیرا معماری به دست آمده قابل استفاده نخواهد بود).

# کاربردهای مستندسازی معماری (Uses of architectural documentation)

- مستندسازی برای یک معماری وابسته به نیازمندی‌ها است (مانند وابستگی معماری یک سیستم به نیازمندهای آن سیستم).
- مستندات معماری باید توسط همه ذینفعان قابل درک باشد.
- مستندات معماری هم تجویزی و هم تشریحی هستند.

۱- ❑ تجویزی (Prescriptive): برای برخی افراد بایدها و نبایدها را نشان می‌دهد.

۲- ❑ تشریحی (Descriptive): برای برخی افراد تشریح می‌کند که تصمیمات اتخاذشده قبلی پیرامون سیستم چه نکاتی هستند (شرح سناریوها برای تعریف نیازمندی .Qtt؟ شرح تاکتیک‌های مورداستفاده و ...).

- همه این توصیفات بیان می‌کند که ذینفعان مختلف مستندات معماری، نیازهای مختلفی دارند.
- این نیازمندی‌ها شامل انواع مختلف اطلاعات و سطوح مختلف اطلاعـات هستند،

❑ به همین دلیل نمی‌توان با یک مستند، معماری را نشان داد.

# ذینفعان سیستم


**آیا معمار می‌تواند تمام خواسته‌های ذینفعان را برآورده سازد؟**

## کاربردهای مستندسازی معماری (Uses of architectural documentation) (ادامه)

- **مستندات معماری** از **دیدگاه ذینفعان مختلف** نوشته می‌شود (ذینفعان مختلف مستندات، نیازهای مختلفی دارند).
- **شناخت ذینفعان** و **نحوهٔ استفاده آن‌ها** از مستندات،
  - ۱ـ ❑ به **سازماندهی مستندات** کمک می‌کند.
  - ۲ـ ❑ به **برقراری ارتباط بین ذینفعان** کمک می‌کند.
- **ذینفعان به مستندسازی نیاز دارند،**
  - ❑ تا از طریق آن بتوانند به **اهداف** خود دست یابند.
- **مستندات معماری** یک **روش کلیدی برای آموزش** افرادی است که **نیاز به دید کلی از سیستم دارند** (مانند: **توسعه‌دهندگان جدید** (new developers)، **کارفرمایان مالی** (funding sponsors)، **بازبینی‌کننده‌های پروژه** (project's visitors) و غیره).

# ذینفعان استفاده‌کننده از معماری

| ذینفعان | نوع استفاده از معماری |
|---|---|
| معمار و مهندسین نیازمندی‌ها | برای بحث و گفتگو در مورد **نیازمندی‌های متضاد** و ایجاد توازن بین آن‌ها. |
| معمار و طراحان بخش‌های تشکیل‌دهنده سیستم | برای حل **رقابت منابع** و **برقراری کارایی** و دیگر هزینه‌های مرتبط با مصرف منابع در زمان اجرا. |
| پیاده‌سازها | برای تهیه محدودیت‌های غیرقابل تغییر در **فعالیت‌های توسعه پائین دست سیستم** (قوانین و قراردادهای برنامه‌نویسی مثل شیوه نام‌گذاری متغیرها). |
| آزمون‌گرها و یکپارچه‌سازها | برای تعیین **رفتار صحیح** در **تست جعبه-سیاه** توسط مولفه‌هایی که باید یکدیگر را کامل کنند. |
| نگهداری‌کننده‌ها | برای تعیین قسمت‌هایی که **تحت تاثیر تغییرات آینده** هستند. |
| طراحان دیگر سیستم‌هایی که با سیستم در ارتباطند | برای **تعیین مجموعه عملیاتی** که برای آنها فراهم شده است و **پروتکل‌هایی** که برای اجرای عملیات نیاز دارند. |

# ذینفعان استفاده‌کننده از معماری (ادامه)

| | |
|---|---|
| متخصص خصوصیات کیفی | برای تهیه مدلی که **ابزارهای تحلیل** مانند شبیه‌سازها و غیره بتوانند از آن استفاده نمایند (مانند ابزارهای تحلیل معیارهای کارآیی معماری مثل بار یا فشار). |
| مدیران | برای **آماده‌سازی تیم توسعه** مطابق با کارهای منتسب شده، **برنامه‌ریزی منابع پروژه** و در جریان بودن از روند کار توسط تیم‌های مختلف. |
| مدیران خط تولید | برای تعیین اینکه آیا **نسخه جدیدی** از **محصول** در نظر گرفته شده است یا خیر. |
| تیم تضمین کیفیت | برای اجرای **آزمون هماهنگی** بین **پیاده‌سازی** و **تجویزهای معماری** (بررسی هماهنگی بین آنچه معماری شده و آنچه قرار است پیاده‌سازی شود). |

# گام بعدی

- اهمیت مستندسازی معماری نرم‌افزار
- انتخاب دیدهای مرتبط
- مستندسازی دید
- مستندسازی بین دیدها
- استفاده از UML

# دیدهای معماری

- مستندسازی معماری، همان مستندسازی دیدهای مرتبط است.
- دید (view)، یک مجموعه منسجم از عناصر معماری و ارتباطات بین آن‌ها را نشان می‌دهد.
- دیدهای گوناگون، اهداف و کاربردهای مختلف را پشتیبانی می‌کنند.
- نیاز به دیدهای مختلفی وجود دارد که هر کدام از یک بُعد، معماری را تشریح کنند.
  - زیرا معماری یک موجودیت پیچیده است و نمی‌توان آن را با یک روش تک بُعدی تشریح کرد
- مستندسازی دیدهای معماری به سه بخش قابل ردیابی تقسیم می‌شود:

۱- انتخاب دیدهای مرتبط

۲- مستندسازی هر یک از دیدها

۳- مستندسازی اطلاعاتی که بر روی بیش از یک دید اِعمال می‌شوند.

# انتخاب دیدهای مرتبط

- دیدهای انتخابی نشان‌دهندهٔ، چیست؟
  - تاکید بر عناصر و روابط آن‌ها از دیدگاه معمار هستند.
- شناسایی اهداف ذینفعان نشان‌دهنده آن است که چه دیدهایی باید در مستند معماری وجود داشته باشند (بنابراین معمار باید ذینفعان را به خوبی شناسایی کند).
- ذینفعان به مستندسازی نیاز دارند تا از طریق آن بتوانند به نیازهای خود دست پیدا کنند.
- همچنین، خصوصیات کیفی خیلی مهم برای ذینفعان،
  - انتخاب دیدهای موردنیاز برای مستندسازی را تحت تاثیر قرار می‌دهند.
- به عنوان مثال: (معماری به عنوان دستورالعمل عملیاتی برای پیاده‌سازها به صورت یک طرح کلی دربردارنده خواسته‌ها و انتظارات ذینفعان است)
  - اگر خصوصیت کیفی قابلیت حمل سیستم برای ذینفعان مهم باشد، باید دید لایه‌ای، انتخاب شود.
  - اگر خصوصیت کیفی کارآیی و قابلیت اطمینان سیستم مهم باشد، باید دید استقرار، انتخاب شود.

# انتخاب دیدهای مرتبط (ادامه)

- سه دسته دید معماری وجود دارد (یعنی معماران نیاز دارند به سه روش مختلف در مورد نرم‌افزار خودشان فکر کنند):

۱- دید ماژول (Module) – سیستم چگونه به صورت یک مجموعه از واحدهای کد یا واحدهای پیاده‌سازی ساخته شود؟

۲- دید مولفه و اتصال (Component-and-Connector) – ساختار عناصر نرم‌افزار که رفتار و تعامل زمان اجرا با یکدیگر دارند چگونه است؟

۳- دید تخصیص (Allocation) – سیستم چگونه به ساختارهای غیرنرم‌افزاری در محیط خودش مرتبط شود؟

- در ادامه، جدولی ارایه شده است که مجموعه‌ای از ذینفعان و انواع دیدهایی که آنها فکر می‌کنند برای اهدافشان مفید است (یعنی از آن دیدها استفاده می‌کنند) را نشان می‌دهد.

# دیدهای مورداستفاده ذینفعان

| | انواع دید | ذینفعان | | | |
|---|---|---|---|---|---|
| | | مدیر پروژه | عضو تیم توسعه | آزمون‌گرها و یکپارچه‌سازها | نگهدارنده‌ها |
| دیدهای ماژول | Decomposition | s | d | – | d |
| | Uses | s | d | d | d |
| | Class | – | d | d | d |
| | Layer | s | d | – | d |
| دیدهای C&C | Various | s | d | s | d |
| دیدهای تخصیص | Deployment | d | s | s | s |
| | Implementation | s | d | s | d |

x = هر چیزی  o = اطلاعات کلی  s = کمی جزئیات  d = جزئیات کامل

# دیدهای مورد استفاده ذینفعان (ادامه)

| | انواع دید | ذینفعان | | | |
|---|---|---|---|---|---|
| | | سازنده خط تولید نرم‌افزار | مشتری | کاربر نهایی | تحلیل‌گر |
| دیدهای ماژول | Decomposition | – | – | – | d |
| | Uses | d | – | – | d |
| | Class | s | – | – | s |
| | Layer | o | – | – | d |
| دید C&C | Various | s | s | s | s |
| دیدهای تخصیص | Deployment | s | o | s | d |
| | Implementation | s | – | – | – |

x = هر چیزی  o = اطلاعات کلی  s = کمی جزئیات  d = جزئیات کامل

# دیدهای مورد استفاده ذینفعان (ادامه)

| | انواع دید | ذینفعان | | |
|---|---|---|---|---|
| | | پشتیبانی‌کننده زیرساخت | ذینفعان جدید | معماران فعلی و آینده |
| دیدهای ماژول | Decomposition | S | x | d |
| | Uses | S | x | d |
| | Class | - | x | d |
| | Layer | s | x | d |
| دید C&C | Various | - | x | d |
| دیدهای تخصیص | Deployment | s | x | d |
| | Implementation | d | x | s |

x = هر چیزی  o = اطلاعات کلی  s = کمی جزئیات  d = جزئیات کامل

# مراحل انتخاب دیدها

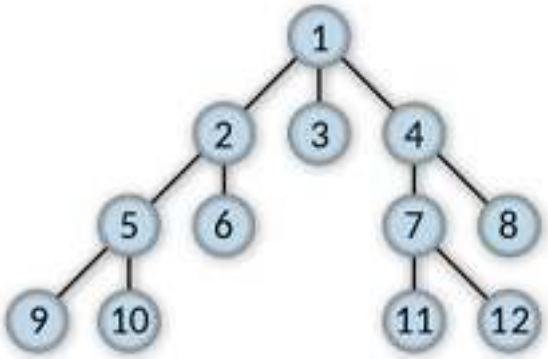
- یک رویه سه مرحله‌ای برای انتخاب دیدهای یک پروژه ارائه می‌شود:
- ۱) فهرستی از دیدهای منتخب را ایجاد نمائید.
  - جدول ذینفعان را ایجاد کنید (ذینفعان را در سطرها قرار دهید)
  - تمام دیدهایی که می‌توانند در پروژه وجود داشته باشند را بنویسید (دیدها در ستون‌ها)
  - خانه‌های جدول را با میزان جزئیات موردنیاز برای هر ذینفع پر کنید.
- ۲) دیدها را ترکیب کنید.
  - برای کاهش تعداد دیدها، آنان‌هایی که نیاز به اطلاعات کلی برای بررسی‌شان وجود دارد، در نظر گرفته و در صورتیکه با دیدهای قوی‌تر برآورده می‌شوند، حذف کنید (در واقع، دیدها حذف نمی‌شوند بلکه با دیدهای دیگر ترکیب می‌شوند).

# مراحل انتخاب دیدها (ادامه)

- ۲) ترکیب دیدها (ادامه)
  - دیدهایی را که می‌توانند ترکیب شوند، با یکدیگر ترکیب کنید.
    - دیدهایی که حاوی اطلاعاتی از دیگر دیدها هستند، برای ترکیب مناسب هستند.
      - برای مثال، دید تجزیه ماژول می‌تواند به راحتی دید پیاده‌سازی، لایه‌بندی و uses را پوشش دهد.
- ۳) دیدها را اولویت‌بندی کنید.
  - بعد از مرحله دوم، یک مجموعه مناسب از دیدها برای نمایش به ذینفعان وجود دارد.
  - دیدها را یکی پس از دیگری کامل (ایجاد) کنید.
  - نیازی نیست دیدها را بترتیب کامل کنید، بلکه می‌توانید به صورت تدریجی آنها را کامل کنید (Breadth-first approach). زیرا:
    1. مدیران شرکت‌ها علاقه‌مندند به‌طور مستمر خروجی معماری را ببینند تا نظرات خود را ارایه نمایند.
    2. در روند ساخت دیدها، ممکن است ذینفعان انتظارات خود را تغییر دهند، لذا با روش ساخت تدریجی دیدها می‌توان مشکلات ناشی از تغییرات را کاهش داد.

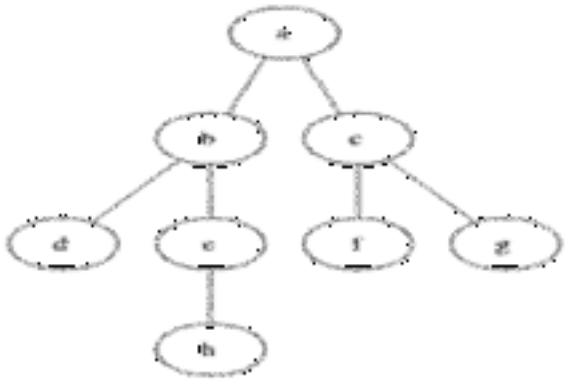
# مراحل انتخاب دیدها (ادامه)

- **Breadth-first approach**




Order in which the nodes are expanded

Animated example of a breadth-first search



# گام بعدی

- اهمیت مستندسازی معماری نرم‌افزار
- انتخاب دیدهای مرتبط
- مستندسازی دید
- مستندسازی بین دیدها
- استفاده از UML

# مستندسازی دید

- استاندارد پذیرفته‌شده‌ای برای مستندسازی دید وجود ندارد.
- در این کتاب، هفت بخش برای مستندسازی دید پیشنهاد می‌شود که تجربه نشان داده است در عمل جواب خوبی داده است.

1. نمایش اولیه دید (Primary Presentation):

- نشان‌دهندهٔ عناصر و ارتباطات بین آن‌ها است. این قسمت دربرگیرنده اطلاعات کلی از سیستم است.
  - معمولاً نمایش اولیه به صورت گرافیکی ارائه می‌شود.
  - می‌توان برای نمایش اولیه از جداول نیز استفاده نمود.
  - نمایش اولیه متنی نیز خلاصه‌ای از مهمترین اطلاعات موجود در هر دید را ارائه می‌کند.
    - مثالی از نمایش اولیه متنی، دید تجزیه ماژول A-7E است (فصل سوم).

# مستندسازی دید (ادامه)

## 2. کاتالوگ عناصر (Element Catalog):

- جزئیات نمایش عناصر و ارتباطات بین آنها در نمایش اولیه و خصوصیات آنها بیان می‌شود.
- مثال: اگر یک نمودار، عناصر A، B و C را نمایش دهد، همراه آن مستندی وجود دارد که هدف و نقش آنها را در سیستم بیان می‌کند.

## 3. نمودار متن (Context Diagram):

- نشان می‌دهد که چگونه دیدهای مرتبط با محیط‌شان بر حسب مفاهیم پایه (واژگان) موجود در دیدها به تصویر کشیده می‌شوند.
- مثال: در دید مولفه و اتصال، نشان داده می‌شود که کدام مولفه با عناصر خارجی و از طریق کدام واسط و پروتکل ارتباط برقرار می‌کند.

# مستندسازی دید (ادامه)

## 4. راهنمای تغییرپذیری (Variability Guide):

- شامل مستنداتی در مورد هر نقطه از تغییرات که در معماری در نظر گرفته شده است (مثلاً یک نقطه تغییر در دید تخصیص، ممکن است شامل شرایطی باشد که تحت آن شرایط، یک عنصر نرم‌افزاری به یک پردازنده خاص تخصیص داده می‌شود).

## 5. پیش‌زمینه معماری (Architecture Background):

- استدلال: بیان علل تصمیمات اتخاذشده در طراحی و بیان علل رد دیگر تصمیمات.
- بیان اینکه چرا طراحی معماری به این شکل است و یک توافق روی تصمیمات انجام شود.
- بیان مفروضات موجود در طراحی معماری (مثلاً قابلیت پشتیبانی یک میلیون بازدیدکننده از سیستم در یک روز).
- نتایج تحلیل بیان می‌کند که چه تغییراتی باید در هنگام تغییر انجام گیرد.

## 6. فرهنگ اصطلاحات (Glossary of Terms):

- توضیحات مختصر در مورد واژگان استفاده شده در طراحی معماری (دید) است.

# مستندسازی دید (ادامه)

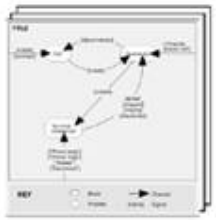
## 7. دیگر اطلاعات (other information):

- اطلاعات این قسمت غیرمعمارانه ولی مفید است.
- متناسب با روش‌های استاندارد یک سازمان خاص، می‌تواند متفاوت باشد.
- این اطلاعات می‌تواند متنوع باشد، مانند اطلاعات مدیریتی مرتبط با مستندات معماری.
- اطلاعاتی از قبیل:
  - ایجادکنندگان مستندات معماری و تاریخ ایجاد،
  - تاریخچه تغییرات،
  - ارجاعات به بخش‌های خاص از مستندات نیازمندی‌ها، برای ایجاد قابلیت ردیابی اطلاعات.
  - و غیره.

# مستندسازی دید (هفت بخش از قالب استاندارد مستندسازی معماری)

**Views**

**Section 1. Primary Presentation of the View**



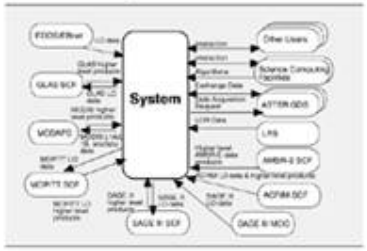
OR

Textual version of the primary presentation

**Section 2. Element Catalog**

Section 2.A Elements and their properties
Section 2.B Relations and their properties
Section 2.C Element interfaces
Section 2.D Element behavior

**Section 3. Context Diagram**




**Section 4. Variability Guide**

**Section 5. Architecture Background**

Section 5.A Design rationale
Section 5.B Analysis of results
Section 5.C Assumptions

**Section 6. Glossary of Terms**

**Section 7. Other Information**

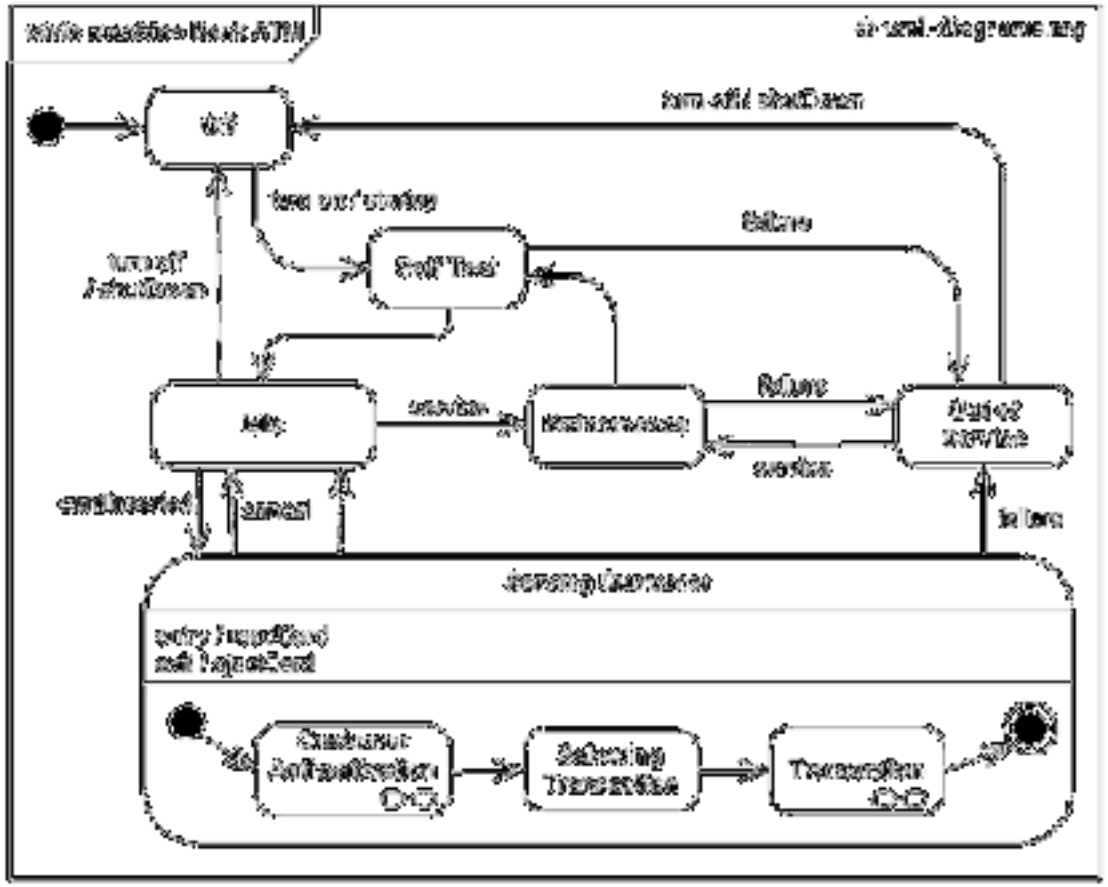
*Source:* Adapted from [Clements 03].

# مستندسازی رفتار (DOCUMENTING BEHAVIOR)

- دیدها، اطلاعات ساختاری سیستم را نشان می‌دهند (برای استدلال پیرامون خصوصیات سیستمی کافی نیستند).
- مستندسازی رفتار، ترتیب تعامل میان عناصر، روش‌های همروندی آنها و وابستگی زمانی تعامل (در یک زمان خاص یا بعد از یک دوره زمانی) آنها را نشان می‌دهد (مثال: استدلال پیرامون بن‌بست به ترتیب تعاملات بستگی دارد).
- رفتار می‌تواند متعلق به یک عنصر یا متعلق به مجموعهٔ از عناصر باشد که با هم هماهنگ کار می‌کنند.
  - رفتاری که مدل می‌شود در واقع به نوع سیستم بستگی دارد.
  - نمونه‌هایی از انتخاب رفتار در سیستم‌های مختلف:
    - در سیستم‌های *RT*، باید در مورد خصوصیات زمانبندی و زمان وقوع رویدادها صحبت کنید.
    - در سیستم بانکداری، ترتیب رویدادها بیشتر از دانستن زمان واقعی وقوع آن‌ها ارزش دارند.
    - در UML، نمودارهای ترتیب و نمودارهای حالت، مثالهایی از تشریح رفتار هستند.

# مستندسازی رفتار (DOCUMENTING BEHAVIOR) (ادامه)

- Behavioral state machine UML diagram example - Bank ATM
- نمودار حالت امکان استدلال در باره کل سیستم را فراهم می‌کند.



# مستندسازی رابط‌های مولفه‌ها

- رابط (واسط) نشان می‌دهد که چگونه دو عنصر وابسته به هم ارتباط دارند.
- با استفاده از تجربیات موفق، الگویی برای مستندسازی رابط‌ها پیشنهاد شده است.

۱ـ شناسه رابط (interface identity):

- وقتی مولفه‌ای دارای چند رابط است، شناسه باعث جداسازی و مشخص کردن آنها می‌شود.

۲ـ منابع فراهم شده (resources provided):

- منابعی که مولفه برای استفاده دیگران فراهم کرده است (قلب یک مستند رابط).
  - نحو منبع (*Resource Syntax*) – چگونه دیگر نرم‌افزارها می‌توانند از این منبع استفاده نمایند.
  - معنای منبع (*Resource Semantics*) – نحوهٔ فراخوانی منبع چگونه است و پس از استفاده چه اتفاقی می‌افتد.
  - محدودیت‌های استفاده از منبع (*Resource Usage Restrictions*) چیست؟ (مقداردهی اولیه به داده قبل از فراخوانی آن و غیره)

# مستندسازی رابطها (ادامه)

- تعریف نوع داده‌ها (Data type definitions):
  - در صورتیکه منبع از نوع داده‌هایی غیر از انواع داده‌های تعریف‌شده توسط زبان برنامه‌نویسی استفاده می‌کند، در این قسمت شرح داده می‌شود.
- تعریف استثناها (Exception definitions):
  - استثناهایی که می‌تواند در هنگام استفاده از واسط منبع رخ دهد (دردسترس نبودن منبع، مشغول بودن منبع، خاموش بودن منبع مثل چاپگر).
- قابلیت تغییرپذیری فراهم‌شده توسط واسط (Variability provided by the interface):
  - آیا واسطها اجازه می‌دهند که مولفه‌ها با استفاده از پارامترها پیکربندی شود (مثلاً پارامترهای یک مولفه، هنگام فراخوانی آن مولفه مقداردهی شود).

# مستندسازی رابطها (ادامه)

- **مشخصات خصوصیات کیفی واسط** (Quality attribute characteristics of the interface):
  - معمار باید مشخص کند، چه خصوصیات کیفی (از قبیل، کارآیی، قابلیت اطمینان) توسط این رابط برای کاربران فراهم می‌شود (مثلاً زمان پاسخ‌دهی مولفه پس از فراخوانی آن توسط واسط، چقدر باشد؟).
- **نیازمندی‌های عناصر** (Elements requirements):
  - چه چیزهایی از دیگر عناصر مورد نیاز است (مثلاً منابع فراهم شده از سایر عناصر برای تکمیل این عنصر- داده‌های تولید شده توسط یک عنصر باید در اختیار عنصر مصرف‌کننده قرار گیرد).
- **دلایل و ویژگی‌های طراحی** (Rationale and design issues):
  - دلایل استفاده از این طراحی و ویژگی‌های طراحی، بیان می‌شود از قبیل،
  - انگیزه طراحی، محدودیت‌ها، توافقات، ملاحظات طراحی، ملاحظات تغییر واسطها در آینده و غیره
- **راهنمای استفاده** (Usage guide)- بیان چگونگی تعاملات بین مولفه‌های مختلف ، بهتر خوانده شدن

# مستندسازی رابطها (قالب پیشنهادی برای مستندسازی رابطها)

**Section 2C. Element Interface Specification**

**Section 2.C.1. Interface identity**
**Section 2.C.2. Resources provided**
Section 2.C.a. Resource syntax
Section 2.C.b. Resource semantics
Section 2.C.c. Resource usage restrictions
**Section 2.C.3. Locally defined data types**
**Section 2.C.4. Exception definitions**
**Section 2.C.5. Variability provided**
**Section 2.C.6. Quality attribute characteristics**
**Section 2.C.7. Element requirements**
**Section 2.C.8. Rationale and design issues**
**Section 2.C.9. Usage guide**

*Source:* Adapted from [Clements 03].

# گام بعدی

- اهمیت مستندسازی معماری نرم‌افزار
- انتخاب دیدهای مرتبط
- مستندسازی دید
- مستندسازی بین دیدها
- استفاده از UML

# مستندسازی بین دیدها

- به‌منظور **کامل کردن مستند دید** باید اطلاعاتی که **به بیش از یک دید یا کل مستند** اعمال می‌شوند، مشخص کرد.
- **مستندسازی بین دیدها حاوی سه جنبه اصلی چرا، چه‌چیز و چگونه** (how, what, why) به شرح زیر است:

**۱- مستند چگونه باید سازماندهی شود تا ذینفع معماری بتواند اطلاعات موردنیاز خود را با سرعت و با قابلیت اطمینان بالا پیدا نماید.**

- این بخش متشکل از **کاتالوگ دید** (View catalog) و **قالب دید** (View template) است.

# مستندسازی بین دیدها (ادامه)

- **کاتالوگ دید** (View catalog):
  - امکانی **برای ذینفعان دیدها** فراهم می‌کند تا اطلاعات خاص **موردنیاز** خود را به راحتی پیدا کنند.
  - در **کاتالوگ دید**، برای هر دید یک **مدخل** وجود دارد که شامل اطلاعات زیر است:
    - نام دید و توصیف اینکه به چه منظوری ایجاد شده است (استفاده می‌شود).
    - توصیفی از خصوصیات، نوع عناصر و نوع ارتباطات استفاده شده در دید.
    - اطلاعات مدیریتی مربوط به مستند دید ازقبیل آخرین نسخه، محل و مالک مستند دید.

# مستندسازی بین دیدها (ادامه)

- **قالب دید** (View template):
  - یک **سازماندهی استاندارد** برای **دید** است.
  - **هدف از قالب دید** آن است که کمک شود:
    - خوانندگان، به راحتی به بخش موردنظر خود دسترسی داشته باشند.
    - نویسندگان، اطلاعات را سازماندهی کنند.
    - معیارهایی ایجاد شود تا باقیمانده کار مشخص شود.

# مستندسازی بین دیدها (ادامه)

## ۲- معماری چه‌چیزی است؟

- مرور کلی سیستم انجام می‌شود،
  - تا ذینفعان با اهداف سیستم آشنا شوند.
- روشی برای مرتبط کردن دیدها با یکدیگر بیان می‌شود (نگاشت بین دیدها).
- فهرستی از مولفه‌ها و محلی که به کار گرفته می‌شوند، ارائه می‌شود.
- فهرستی از واژه‌نامه پروژه و معنی هرکدام بیان می‌شود.

# مستندسازی بین دیدها (ادامه)

## ۳- چرا معماری این‌گونه طراحی شده است؟

۱- ■ اشاره به تصمیمات اتخاذشده در کل سیستم (چرا این خصوصیات کیفی؟ چرا این الگوهای معماری؟ و ...)،

□ برای برآوردن نیازمندی‌ها و محدودیت‌ها.

۲- ■ تاثیرات معماری در هنگام،

□ افزودن نیازمندی جدید یا تغییر یکی از نیازمندی‌های فعلی.

۳. ■ محدودیت‌هایی که توسعه‌دهنده در پیاده‌سازی خواهد داشت.

■ بیان علل دیگر تصمیماتی که رد شده‌اند.

# مستندسازی بین دیدها (ادامه)

- خلاصه‌ای از مستند بین دیدها:

**Documentation across Views**

**How the document is organized:**
- 1.1 View catalog
- 1.2 View template

**What the architecture is:**
- 2.1 System overview
- 2.2 Mapping between views
- 2.3 List of elements and where they appear
- 2.4 Project glossary

**Why the architecture is the way it is:**
- 3.1 Rationale

*Source:* Adapted from [Clements 03].

# گام بعدی

- اهمیت مستندسازی معماری نرم‌افزار
- انتخاب دیدهای مرتبط
- مستندسازی دید
- مستندسازی بین دیدها
- استفاده از UML

# دیدهای موجود در UML

## مقدمه:

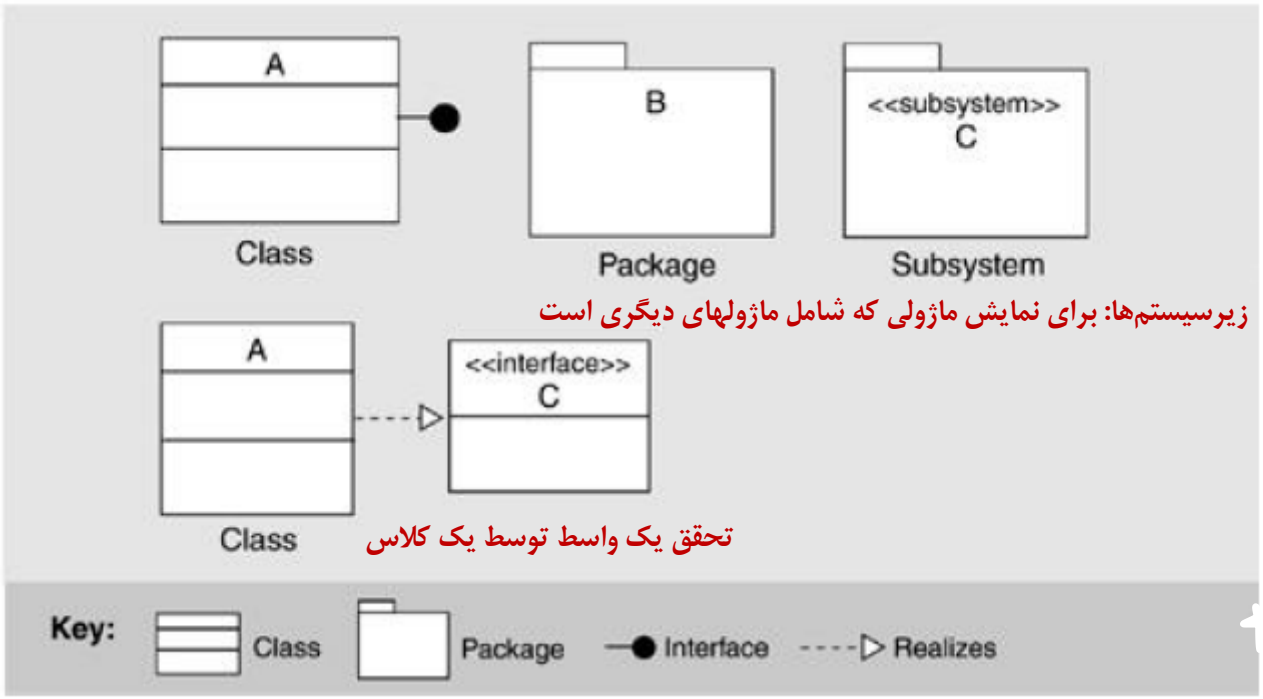
- معماری بیان می‌کند که چه چیزی برای یک سیستم نرم‌افزاری ضروری است.
- ماهیت معماری معمولاً مستقل از زبان و نمادها است.
- با این وجود، UML به‌عنوان نماد استاندارد برای مستندسازی معماری مورد استفاده قرار می‌شود.
- از UML، برای نمایش اولیه، تشریح رفتار مولفه‌ها یا گروهی از مولفه‌ها می‌توان استفاده کرد.

# دیدهای موجود در UML (ادامه)

- دید ماژول (Module Views):
  - واسطها ( Interfaces)
  - تعدادی ماژول ( Modules) با رابطه‌های زیر بین آنها:
    - رابطه تجمیع (تعمیم «Generalization»)
    - رابطه وابستگی ( Dependency)
- دید مولفه و اتصال (Component-and-Connector Views)
  - Components
  - Interfaces
  - Connectors
  - Systems
- دید تخصیص (Allocation views)

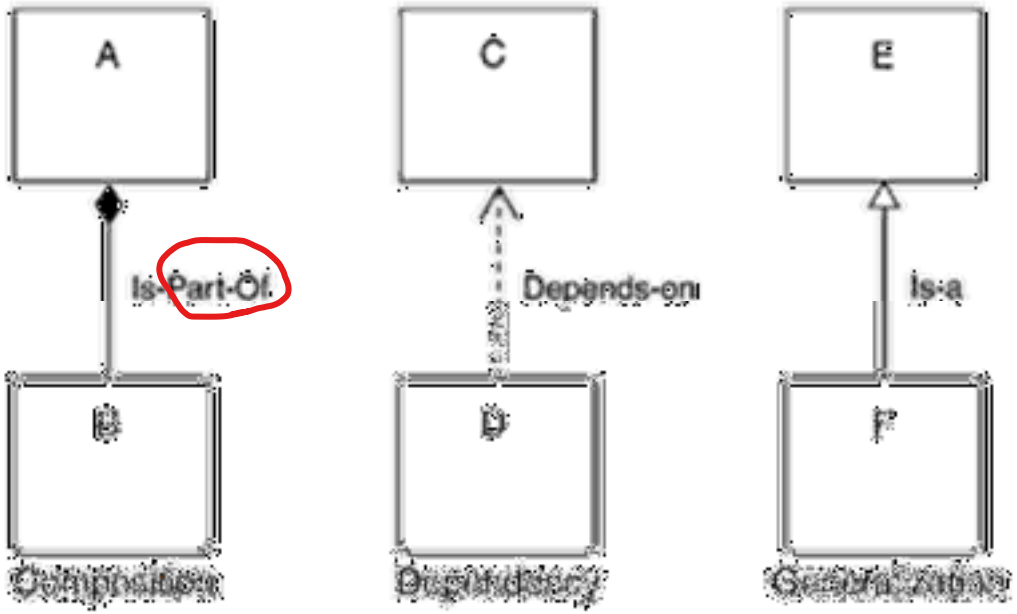
# دید ماژول (Module View)

- ماژول یک واحد کد یا یک واحد پیاده‌سازی است.
- دید ماژول، تعدادی از ماژولها با واسطهای مربوطه و دیگر رابطه‌های موجود بین آنها است.
- نمونه‌هایی از نمادهای ماژول در UML:



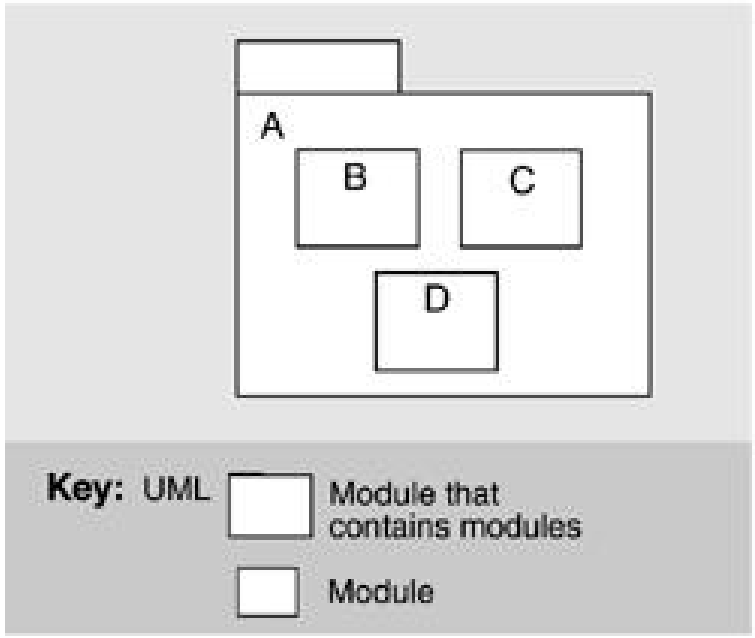
# دید ماژول (Module View) (ادامه)

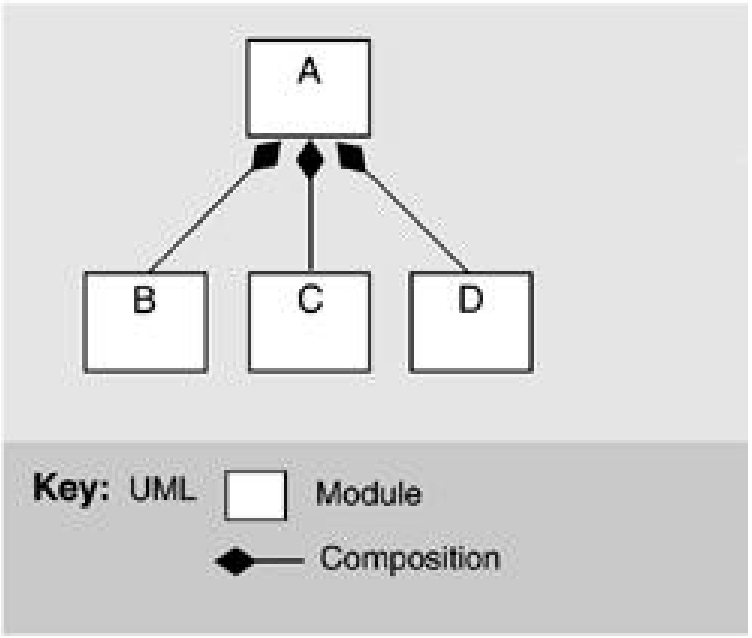
- نمونه‌هایی از **رابطه‌های** موجود در **دید ماژول**:
  - رابطه «بخشی از» **Is-Part-Of** : ماژول B بخشی از ماژول A است (دید تجزیه ماژول متکی بر این رابطه است).
  - رابطه «وابستگی» **Depends-on** : ماژول D وابسته به ماژول C است (دید uses ماژول متکی بر این رابطه است).
  - رابطه تعمیم یا «نوعی از» **Is-a** : ماژول F نوعی از ماژول E است (دید class ماژول متکی بر این رابطه است).
    - رابطه تعمیم برای نشان دادن یک رابطه وراثتی بین کلاس‌ها استفاده می‌شوند.



# دید ماژول (Module View)(ادامه)

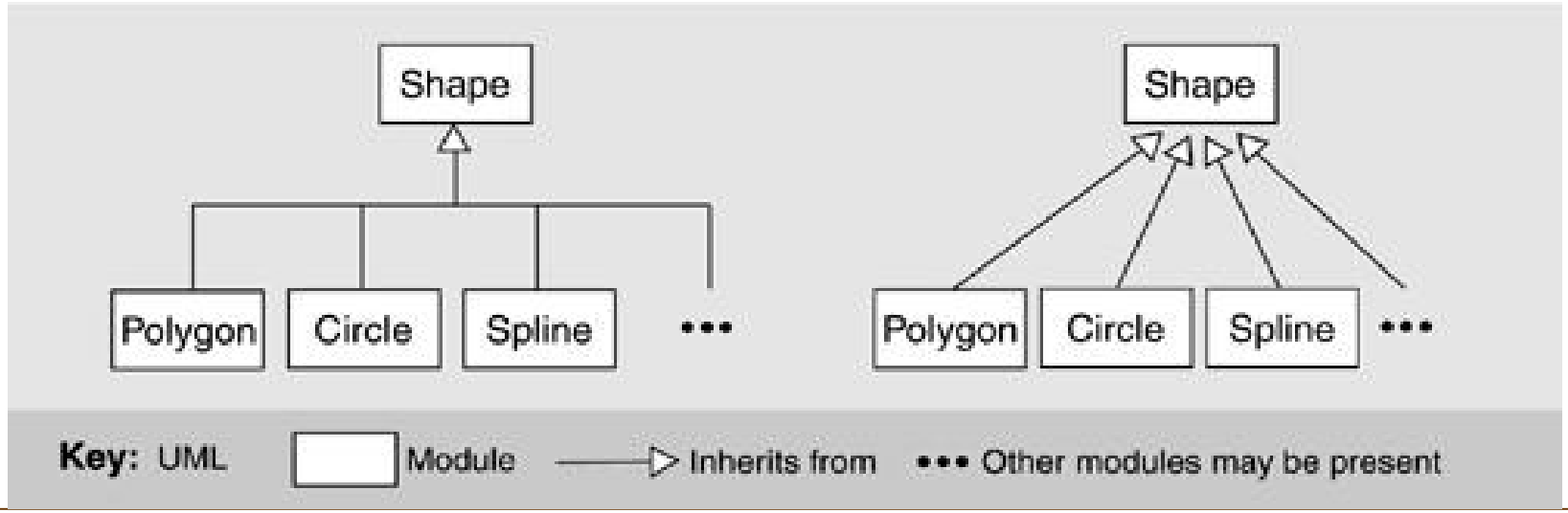
- **رابطه تجمیع** ( Aggregation) را می‌توان به صورت زیر در UML نشان داد:
  - ماژول‌ها می‌توانند تو در تو باشند (سمت چپ)
  - با استفاده از ترکیب (Composition) با یکدیگر ارتباط داشته باشند (سمت راست)
    - در UML، ترکیب شکلی از تجمیع است.



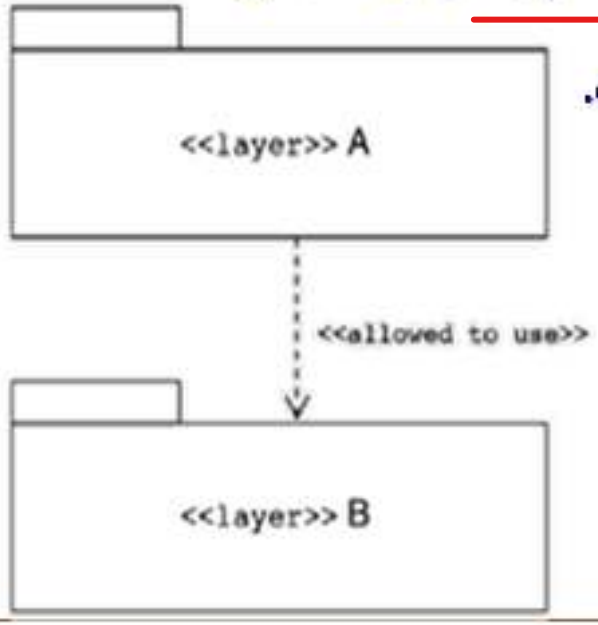
# دید ماژول (Module View)(ادامه)

- مستندسازی رابطه تعمیم (Generalization) توسط UML: ارث‌بری
  - در UML، اگر ماژول‌ها به صورت نمودار کلاس نشان داده شوند، رابطه تعمیم نقش مهمی ایفا می‌کند.
  - از نظر معنایی دو نمودار زیر یکی هستند.
  - ماژول Shape، پدر و بقیه ماژول‌ها فرزند (زیر کلاس) هستند.
  - ماژول Shape، حالت عمومی دارد و سایر ماژول‌ها یک نسخه خاص از آن هستند.



# دید ماژول (Module View) (ادامه)

- رابطه وابستگی:
- مهمترین محل نمایش «رابطه وابستگی»، در لایه‌ها است.
- UML نماد پایه‌ای خاص داخلی برای نمایش لایه‌ها ندارد.
  - می‌توان با استفاده از مفهوم *Package* وابستگی لایه‌ها را به نوعی نمایش داد.
  - «رابطه وابستگی» در این حالت «اجازه استفاده» است.



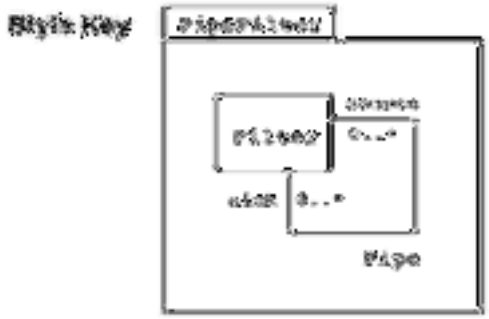
# دید مولفه و اتصال ( Component-and-Connectors)

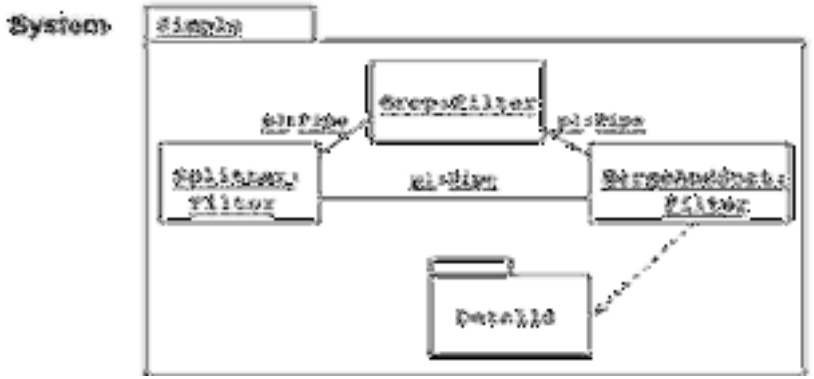
- راهبرد مشخصی برای نمایش این دید در UML وجود ندارد.
- اما، روش‌هایی برای نمایش آن وجود دارد که هر یک دارای مزایا و معایب خاص خود هستند.

  - یک روش طبیعی برای نمایش انواع «دید مولفه و اتصال» استفاده از مفهوم کلاس در UML است.

# دید مولفه و اتصال ( Component-and-Connectors) (ادامه)

- شکل زیر ایده عمومی این روش را نشان می‌دهد که از الگوی لوله و فیلتر استفاده شده است.
  - نوع معماری فیلتر به عنوان کلاس *Filter* در نظر گرفته شده است.
  - نمونه‌های فیلترها (مانند *splitter*) به عنوان اشیاء فیلتر نمایش داده شده‌اند.



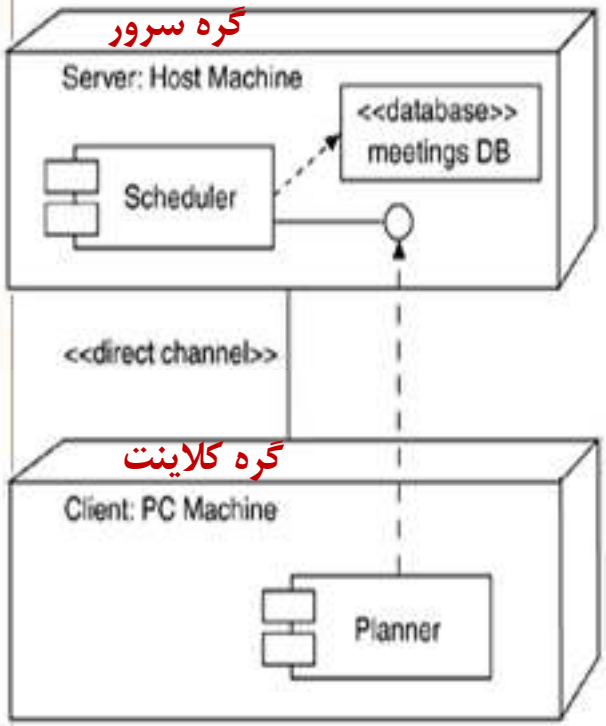


# دید مولفه و اتصال ( Component-and-Connectors) (ادامه)

- **مطالعه بیشتر:**
  - راه‌های نمایش رابطهای مولفه‌ها،
  - راه‌های نمایش اتصالات مولفه‌ها،
  - راه‌های نمایش سیستم‌ها را بیان نمایید.
- **مرجع: کتاب Len Bass**

# دیدهای تخصیص (Allocation Views)

- نمودار استقرار در UML، یک گراف از گره‌های مرتبطشده بوسیله رابطه انجمنی هستند.
- رابطه انجمنی، یک رابطه معنایی بین کلاس‌ها است. اطلاع از صفات و رفتار عمومی یکدیگر (رابطه دو طرفه).
- این نمودار برای نمایش چگونگی تخصیص عناصر در زمان اجرا مورد استفاده قرار می‌گیرد.
- این نمودار، اجرای مولفه‌ها (اشیاء) بر روی گره‌ها را نشان می‌دهد.
- گره، یک شیء فیزیکی زمان اجرا است که دارای حافظه و قابلیت پردازشی است که بوسیله رابطه‌های انجمنی (کانال، شبکه) به‌هم متصل
- مولفه‌ها می‌توانند بوسیله خط‌چین پیکان (رابطه وابستگی) به یکدیگر متصل شوند (استفاده یک مولفه از سرویس‌های مولفه دیگر).



# پایان فصل نهم