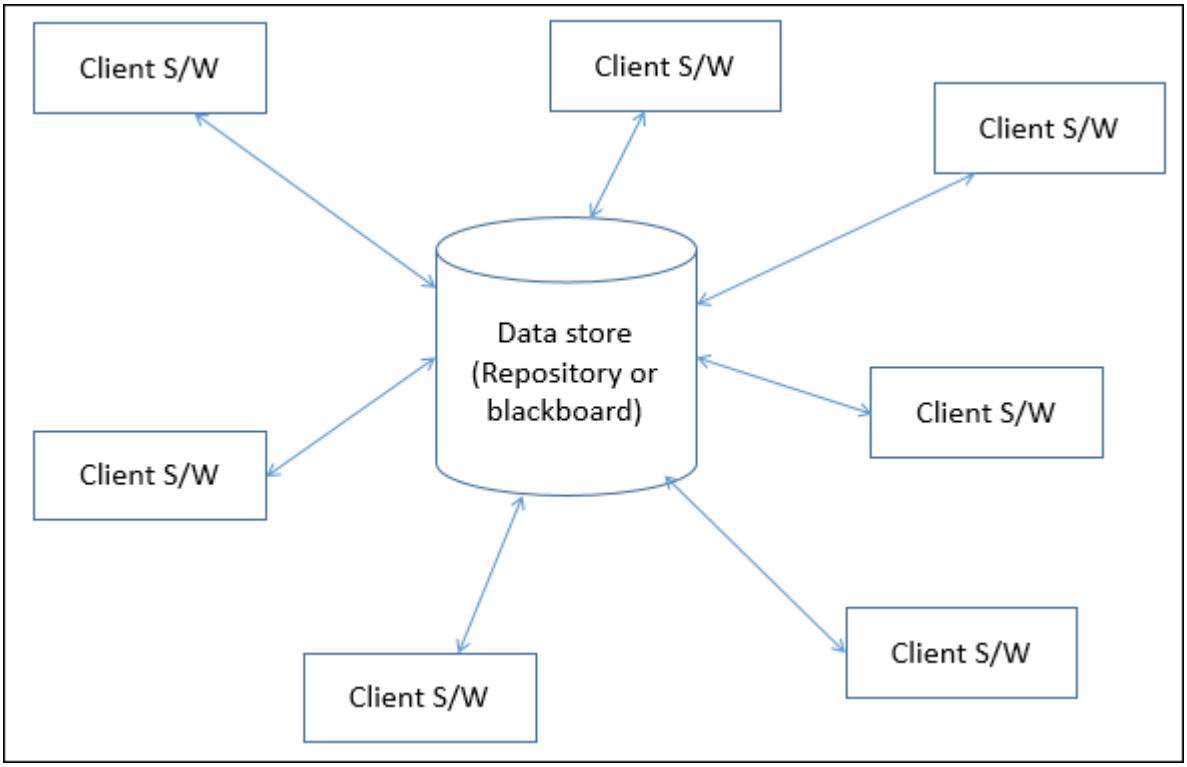
توضیحات سیستم:

موسسه‌ها یا افراد محتوای آموزشی خود را در اختیار سیستم قرار می‌دهند و سیستم پس از دسته‌بندی آن‌ها را در معرض نمایش قرار می‌دهند.
افراد می‌توانند پس از ثبت‌نام به صورت رایگان (۲۰۰ دقیقه یا ۱۰ روز، هر کدام که زودتر تمام شود) از محتوای آموزشی موجود استفاده کنند. در صورت پایان مهلت استفاده رایگان، با خرید اشتراک (ماهانه یا سالانه) می‌توانند به استفاده خود ادامه دهند.

کاربران می‌توانند در قسمت مدیریت courseهای انتخاب شده خود را مشاهده و با انجام تست از میزان مهارت خود اطلاع پیدا کنند. سیستم می‌تواند بر اساس courseهای انتخاب شده کاربر به او محتوای جدید پیشنهاد دهد.

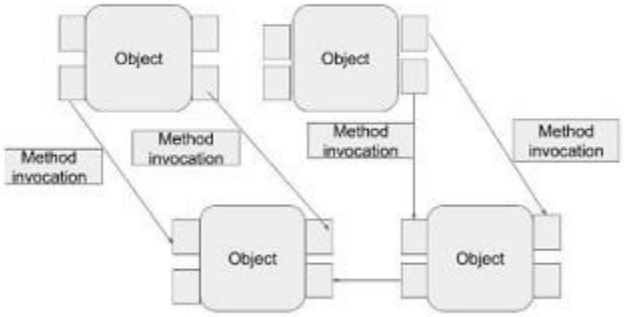
در قسمت notes کاربران می‌توانند یادداشت‌های خود را قرار دهند و در قسمت paths، courseهای به هم مرتبط (مثلاً front-end development) را مشاهده کنند.

1) Data-centered: در این استایل ما به یک data store نیاز داریم، که component های مستقل سیستم با توجه به الگویمان با data store ارتباط برقرار می‌کند. از مزایای این استایل قابلیت scalability است زیرا client ها مستقل هستند و از معایب آن این است که client ها به data store وابسته هستند و بدون آن نمی‌توانند کاری انجام دهند.



با توجه به نیازمندی‌های استخراج شده سیستم، وابستگی اجزا به database و استقلال client ها برداشت می‌شود و به همین دلیل data-centered یکی از کاندیدهای معماری سیستم است.

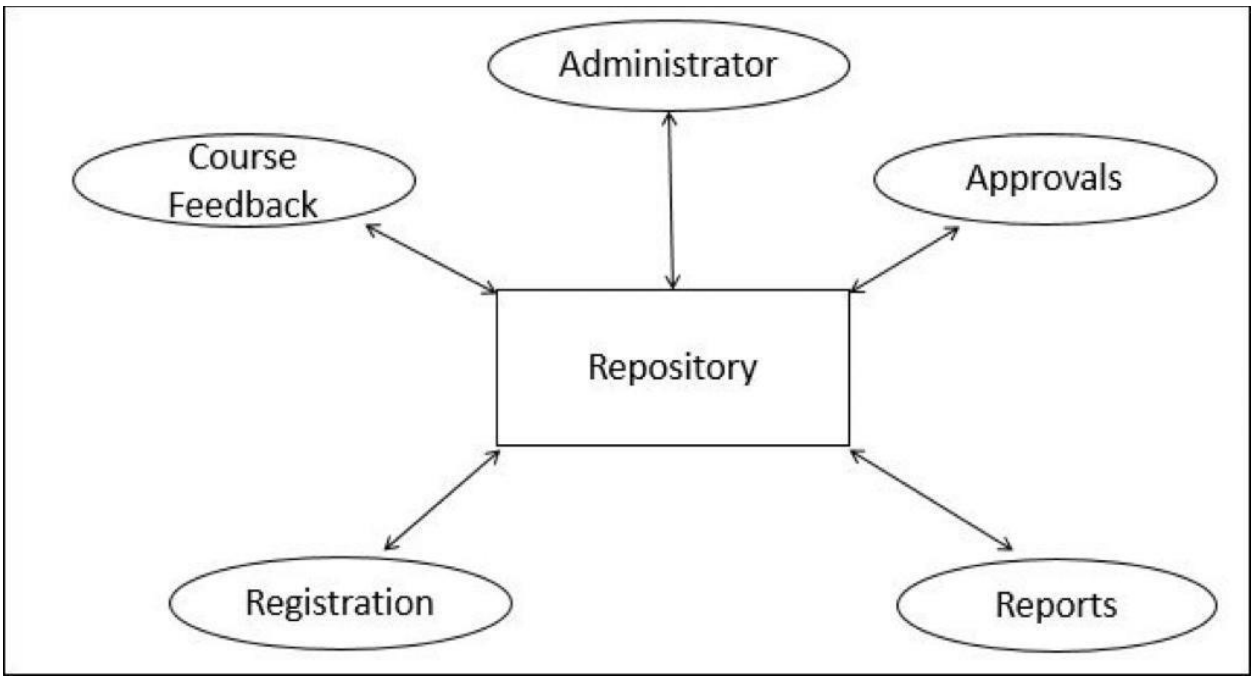
2) Object-oriented: یک شیوه طراحی است براساس تقسیم وظایف برای یک کاربرد یا سیستم به صورت اشیاء قابل استفاده مجدد و خودبسنده. این شیوه سیستم نرم‌افزاری را به عنوان مجموعه‌ای از گونه‌ها به نام object نگاه می‌کند. این استایل بر اساس مدل کردن اشیاء دنیای واقعی است.



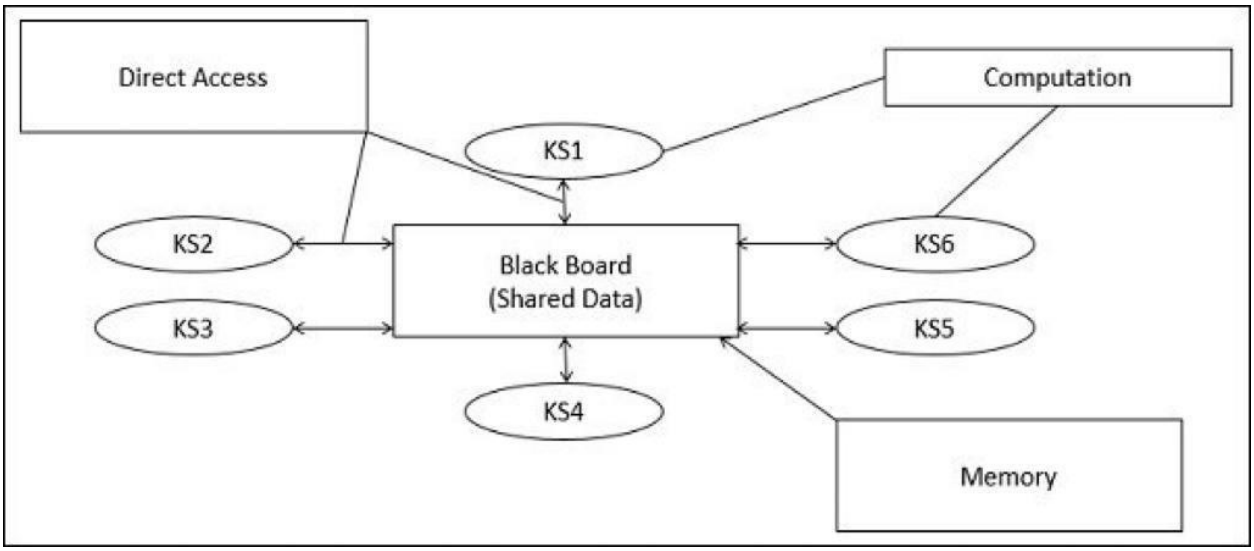
از مزایای این استایل می‌توان به قابل فهم و آزمایش بودن، قابلیت استفاده مجدد، قابلیت تعمیم و نگهداری اشاره کرد. از معایب این استایل می‌توان به مناسب نبودن برای استفاده در سیستم‌های بزرگ اشاره کرد.
به دلیل اینکه این استایل بر اساس مدل کردن اشیاء دنیای واقعی است و اینکه سیستم ما با اشیاء دنیای واقعی سر و کار دارد، به همین دلیل، object-oriented یکی از کاندیدهای معماری سیستم است.

data-centered:

Repository: در این pattern یک repository به صورت passive قرار دارد؛ یعنی به تنهایی کاری انجام نمی‌دهد و client های متصل به آن به صورت active درخواست می‌دهند. به عنوان مثال insert



Blackboard: در این pattern یک Blackboard به صورت active قرار دارد؛ یعنی اجزای متصل به آن بر اساس تغییرات blackboard فعالیت می‌کنند. مثال sound/image recognition



به نوعی می‌توان سیستم را به سه بخش تقسیم کرد.بخش اول وظیفه دریافت اطلاعات از بخش مدیریت اطلاعات را دارد. بخش دوم وظیفه نشان دادن منو بر اساس اطلاعات دریافتی را بر عهده دارد. بخش سوم قابلیت انتخاب از منو را به کاربر می‌دهد.
بر اساس این مسائل الگوی معماری MVC(Model-View-Controller) مناسب است اما برای پردازش درخواست‌های چند کاربر به صورت همزمان لازم است که از مدل PAC(Presentation-Abstraction-Control) استفاده کنیم.

مزایا:

- مناسب ترین روش برای پردازش های توزیع شده در یک شبکه با تعداد کاربران زیاد
- سهولت در امر پیاده سازی
- نسبت دهی مستقیم رابط کاربر با منابع تامین داده ها

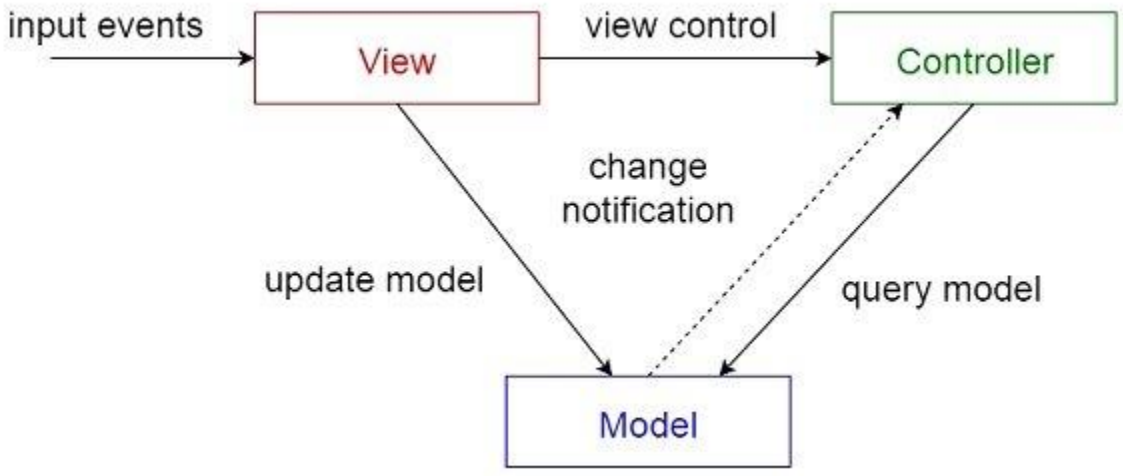
معایب:

- کاهش کارائی برنامه همزمان با افزایش تعداد کاربران همزمان

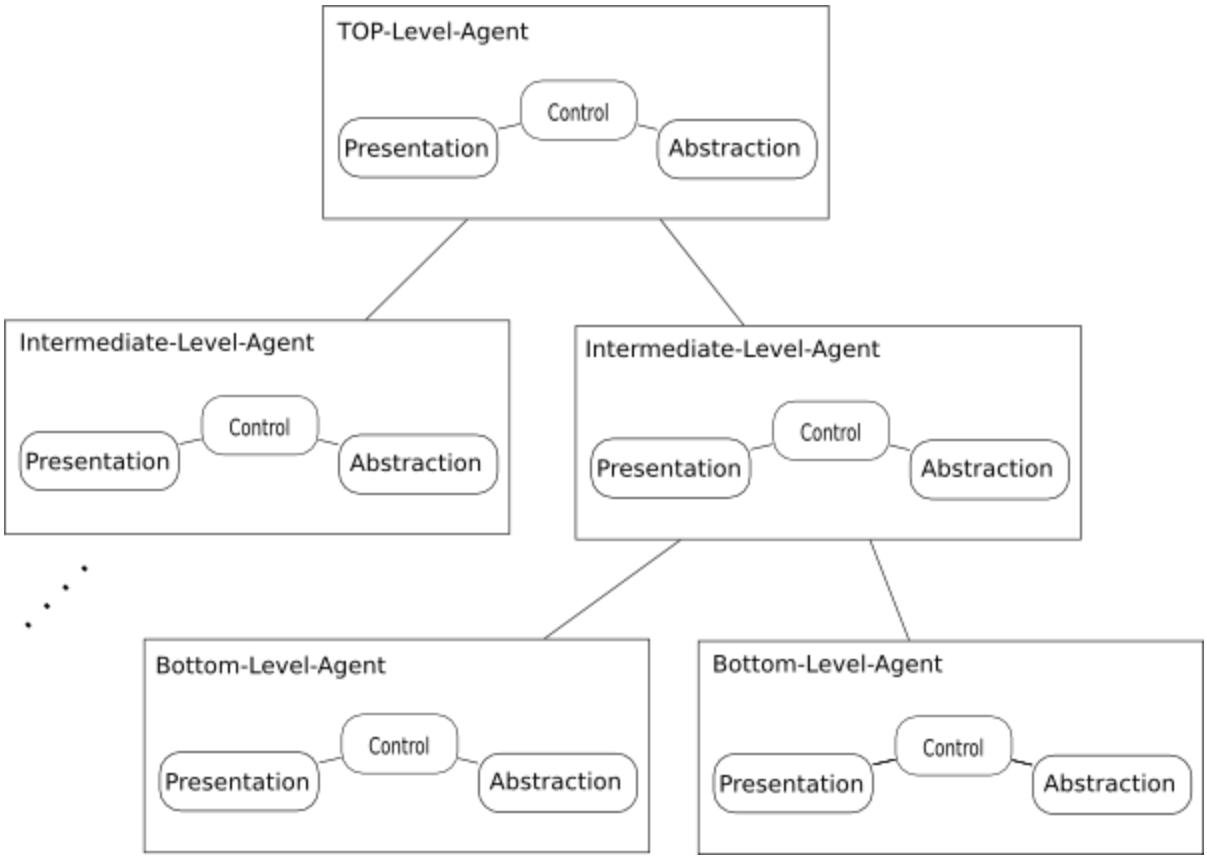
MVC:

این الگو که اغلب موارد با واژه MVC از آن نام‌برده می‌شود، یک برنامه تعاملی را به سه بخش زیر تقسیم می‌کند.

1. Model: که شامل قابلیت‌های اصلی برنامه و داده‌ها است .
2. View: وظیفه این بخش نشان دادن اطلاعات به کاربر است. در برخی از برنامه‌ها ویو نقشی بیش از نشان دادن اطلاعات بر عهده دارد.
3. Controller: این بخش مدیریت داده‌های ورودی را که از سوی کاربران وارد می‌شود، برعهده دارد. همچنین وظیفه برقراری ارتباط میان model و view نیز بر عهده مولفه کنترلر است.



PAC: مانند MVC با این تفاوت که هر کدام از برنامه‌ها(agent) از طریق بخش control با یکدیگر تعامل دارند؛ تفاوت دیگر اینکه presentation و abstraction کپسوله هستند که امکان multithreading را ممکن می‌سازد.



Data-centered Pattern = blackboard
Object oriented Pattern = PAC

Component diagram: