التصميم

د.محمد الأحمد

مرحلة التصميم

هي مرحلة خلاقة، تعني الانتقال من فضاء المشكلة (problem domain) إلى فضاء الحل (solution Space) عن طريق هذه المرحلة حيث نحدد شكل النظام بحيث يلبي المتطلبات الوظيفية ويحترم المتطلبات غير الوظيفية.

تشمل مرحلتين:

1. التصميم المعماري: أول خطوة للتصميم دخلها هو وثيقة SRS مع المخططات التي تحويها ويدرس كيف ننظر للنظام ككل مكون من أنظمة جزئية متفاعلة مع بعضها البعض.
2. التصميم التفصيلي: يشرح التصميم التفصيلي لكل مكون من حيث الخوارزميات والواجهات وقاعدة البيانات.

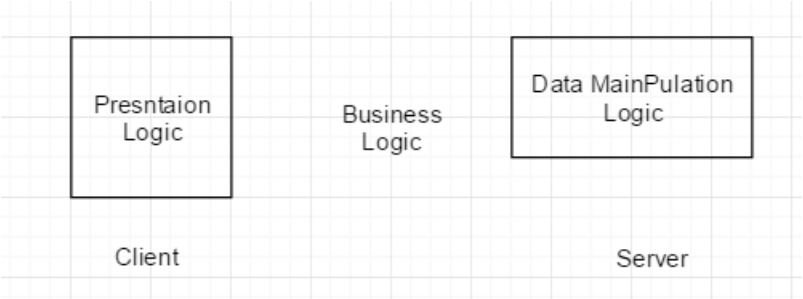
ناتج هذه المرحلة هو Design Document في هذه المحاضرة سندرس التصميم المعماري:

وهو يدرس هيكلية النظام (Structure)

* مبادئ تصميمية أساسية:

1. Modularity: (التجزئة) وتعني مقدار تقسيم النظام فلا يكون عبارة عن كتلة واحدة وإنما أجزاء، قديماً كانت النظم عبارة عن كتلة واحدة عند حدوث أي خطا في خدمة يتوقف النظام وأي تعديل يجب التعديل في جميع الأجزاء.

ثم بدأت النماذج المعمارية بالظهور، بداية بمعمارية [2-tiers] أو ما يسمى (Client-Server) حيث يكون النظام عبارة من جزئين server أو مخدم أساسي يتواصل معه مكون ثاني وهو Client عندما نجري التعديلات تكون فقط على الServer ولا يوجد داع للتعديل لدى ال (client).



ينقسم 2-tiers لنوعين:

1) Fat-Client: طبقة ال Business تقع في قسم الClient وقد تسبب مشاكل في اثناء التعديل.

2) Thin-Client: طبقة الBusiness تقع في قسم ال server وهي الأفضل من حيث الأداء.

بعد فترة ظهر ما يسمى 3-tiers والذي يضيف طبقة لـ Business Logic وهي طبقة Application Server نتيجة التضخم والتوسع ظهرت تطبيقات من نوع n-tiers حيث أن الطبقة الوسطى يمكن أن تتوزع بعدة طبقات حسب الخدمة، مثال: تطبيق لفندق يقدم خدمات الحجز عندما نريد إضافة خدمات إضافية مثل سعر صرف الدولار يتصل بطبقة أخرى مثل البنك، وهذا أفضل كلما زادت عدد الطبقات زادت سهولة الصيانة وإصلاح الأخطاء.

الهدف من هذه الطبقات:

عزل طبقة الزبون عن طبقة قاعدة المعطيات عبر طبقة Business Logic بروتوكولات معينة.

1) Coupling: تقيس درجة ترابط ال Modules مع بعضها البعض.

2) Cohesion: يقيس درجة استقلالية الModules.

العلاقات بينها:

هما مفهومان متعاكسان عندما Coupling عالٍ يكون هناك عدد كبير من المكونات ويوجد ترابط بينهم وعندها ال Cohesion يكون منخفض والاستقلالية تكون منخفضة، أما عندما نقلل عدد المكونات ونزيد الخدمات التي يقدمها كل مكون يقل ال Coupling ويزيد ال Cohesion وكلاهما معاً يحددان ال.Modularity

* كيف نختار ال Coupling وال Cohesion؟

غالباً نفضل أن تكون قيمتهما متوسطة لكن في حالات معينة يتغير ذلك، مثلاً لو أننا نريد سهولة صيانة نختار Coupling عالٍ لأن تحديد الخطأ يكون أسهل، لو أردنا أداء عالٍ نختار Cohesion عالٍ وبالتالي المتطلبات غير الوظيفية هي ما يحدد شكل النظام.

تعريف: عملية تصميم لتحديد الأنظمة الجزئية التي تشكل النظام وكيف تتواصل هذه الأنظمة مع بعضها البعض وهذا هو Architectural Design خرج هذه العملية هو وصف المعمارية البرمجية ((Software Architecture.

أهمية المعمارية البرمجية: في مرحلة الاختبار نبدأ باختبار الواحدة (Unit testing) ثم اختبار التكامل للنظام (Integration testing) لنتأكد من تواصل الأنظمة الجزئية مع بعضها عندها نحتاج للعودة إلى وثيقة المعمارية البرمجية لكي تخبرنا بنية النظام والأنظمة الجزئية التي تعمل مع بعضها.

بعض خواص التصميم المعماري:

* أول مرحلة من عملية التصميم.
* الصلة بين مرحلة الSpecification وبقية مراحل التصميم.
* تتم على التوازي مع نشاطات ال Specification.
* تحدد الأجزاء الأساسية لمكونات النظام وتواصلها.
* تعتمد بشكل أساسي على المتطلبات الوظيفية وغير الوظيفية وقيود الHard ware مثل وجود قاعدة معطيات جاهزة مسبقاً.

أنواعها:

هناك نوعان من التصميم المعماري:

1. Large: تستخدم في النظم الكبيرة الضخمة، نعتمد مبدأ الصقل المتدرج أي ان معمارية لا تتم بخطوة واحدة وإنما تحتاج للتجريد، نبدأ بالتفكير بالأنظمة الجزئية الأساسية ثم تنتقل للأنظمة الجزئية التي تشكل الأنظمة الأساسية.

2. Small: تستخدم في إيجاد بنية الأجزاء الصغيرة من النظام (الانظمة الجزئية البسيطة).

الهدف منها:

1. التواصل مع Stakeholders: عند الانتهاء من مرحلة التصميم المعماري نعرض المخططات على الزبون على شكل Box & line Diagram ونأخذ feedback من الزبون.
2. التواصل مع محلل النظام: لتحسين البنية المعمارية.
3. Large Scale Reuse: اختيار المعمارية المناسبة لهذا النظام والتعديل عليها لتتوافق مع المتطلبات والخدمات.

Architectural Representations

يمكن أن تستخدم Block Diagram على شكل Box & line Diagram لكنه لا يعطي معلومات وتفاصيل عن كيفية التواصل بين الأنظمة الجزئية فهو مفيد للتواصل مع الزبون وعملية التخطيط للمشروع كونه مخطط مجرد (abstract).

Use of Architectural Models:

- يعتمد على نمط النظام لتحديد المعمارية مثال: كل أنواع الكومبايلرات تستخدم pipe & filter ، كل أنواع نظم المعلومات التي تحتاج أمان تستخدم loggers
- بالإضافة إلى المتطلبات الوظيفية وغير الوظيفية نأخذ الHard ware بعين الاعتبار من حيث البيئة الفيزيائية والمعدات.

مجموعة من الأسئلة أثناء تحديد النموذج المعماري المناسب:

- هل يوجد بيئة معمارية عامة للنظام؟ (بيئة معمارية).
- كيف سيوزع النظام؟
- ما هو النمط المعماري المناسب؟
- ما هي منهجية هيكلة النظام؟
- ما هي الModules المكونة للنظام؟ وكيف سيتم تجزئته؟
- ما هي استراتيجية التحكم بعملية التوزيع لأقسام؟
- كيف سيتم تقييم النظام؟
- كيف سيتم إنشاء ال documentation؟

Architectural Reuse

نعود للـ Coupling والـ Cohesion ليساعدنا على معرفة كيفية النظام إلى أنظمة جزئية.

- **Performance**: نريد أداء عالٍ وتواصل أقل عندها نستخدم **large rather** أكثر من **fine-grain** من أجل المكونات.

ملاحظة: **Large rather** تعني حبيبات كبيرة أي أجزاء النظام كبيرة والـ**Cohesion** عالٍ ← **Coupling** منخفض.

Fine-grain : تعني حبيبات صغيرة أي أجزاء النظام صغيرة والـ **Coupling** عالٍ ← **Cohesion** منخفض.

- **Security**: فوراً نفكر بطريقة الطبقات (**layers**) والأجزاء الحساسة تتوضع في الطبقات الداخلية مثال: **OSE**
- **Safety**: تشابه الـ **Security** الطبقات التي تتطلب أمان تتوضع في الأنظمة الجزئية الصغيرة (الطبقات الداخلية).
- **Availability**: الإتاحية أي يكون النظام متاح دوماً وذلك عن طريق تصميم أجزاء بديلة فلو تعطل جزء يكون له بديل يقدم خدماته.
- **Maintainability**: سهولة الصيانة أي سهولة تحديد مكان الخطأ وصيانته عبر **Fine-grain** و**Coupling** عالٍ.

4+1 view model of Software Architecture

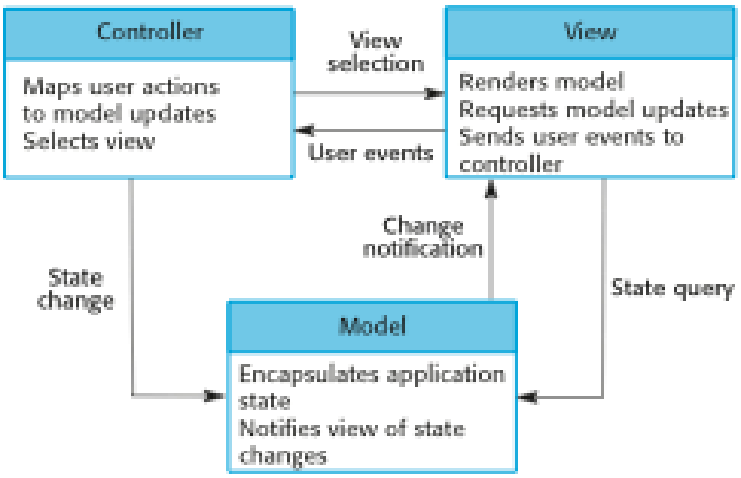
1. **Logical view** : (نظرة منطقية) تتعامل مع الـ**objects** وكيف تتواصل مع بعضها البعض.
2. **Process view** : (نظرة إجرائية) كيف تتم الخدمات وكيف تتواصل المكونات لتقديم الخدمات.
3. **Development view** : (نظرة تطوير) كيف يتم تجميع النظام للتطوير.
4. **Physical view** : (نظرة فيزيائية) تهتم بالبنية الشبكية للنظام وأجزاء الـ**Hardware** .
5. **Rerated Use Case & Scenarios** حيث أن الخدمات المتشابهة تتجمع في طبقة واحدة.

Architecture Patterns

أنماط جاهزة لكل منها ميزاتها أما **Design Patterns** فهي حلول لمشاكل مكررة قد تواجهنا أثناء تصميم الأنظمة الجزئية.

1. **Module View Controller** :**MVC**

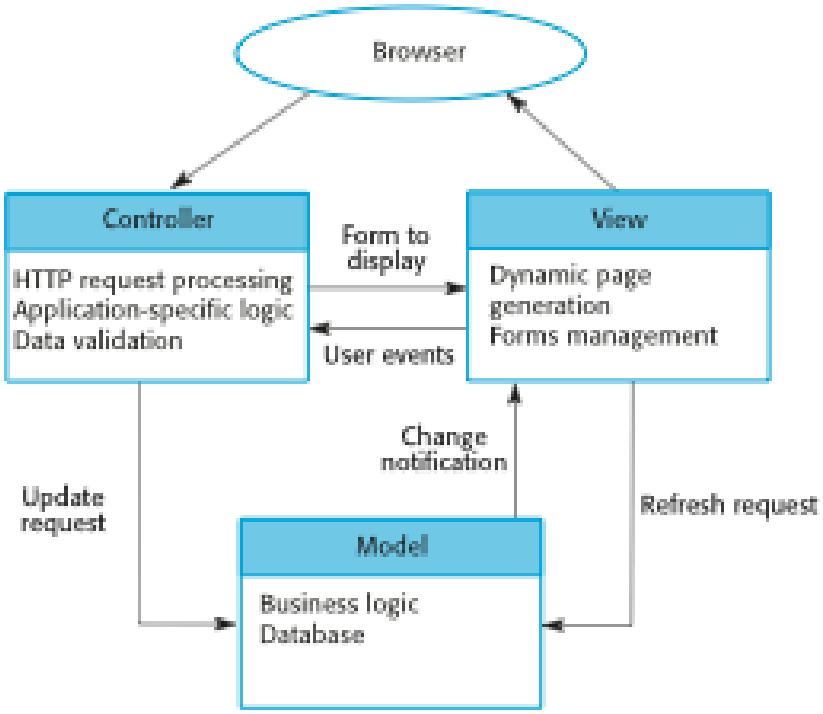
ظهرت في السبعينيات، تستخدم لتطوير جميع أنواع الالعاب تستخدم لعزل طبقة الـ**presentation** عن طبقة الـ**Data** كما في تطبيقات الويب ولكن يمكن أن يكون بينهما تواصل كما في الألعاب.



ملاحظة حول المخطط: في تطبيقات الويب ممنوع السهم بين controllerوال view لكن أحياناً يكون موجود وقد تكون عملية Refresh.

استخداماتها:

تطبيقات الويب وقد يكون التطبيق مكون من أكثر من MVC حسب ضخامة التطبيق.



ملاحظة: الPresentation هو ال logic الذي يحول صفحة الHtml إلى واجهة عرض للمستخدم وليس ما يتم عرضه على المتصفح.

2.Layered Architecture:

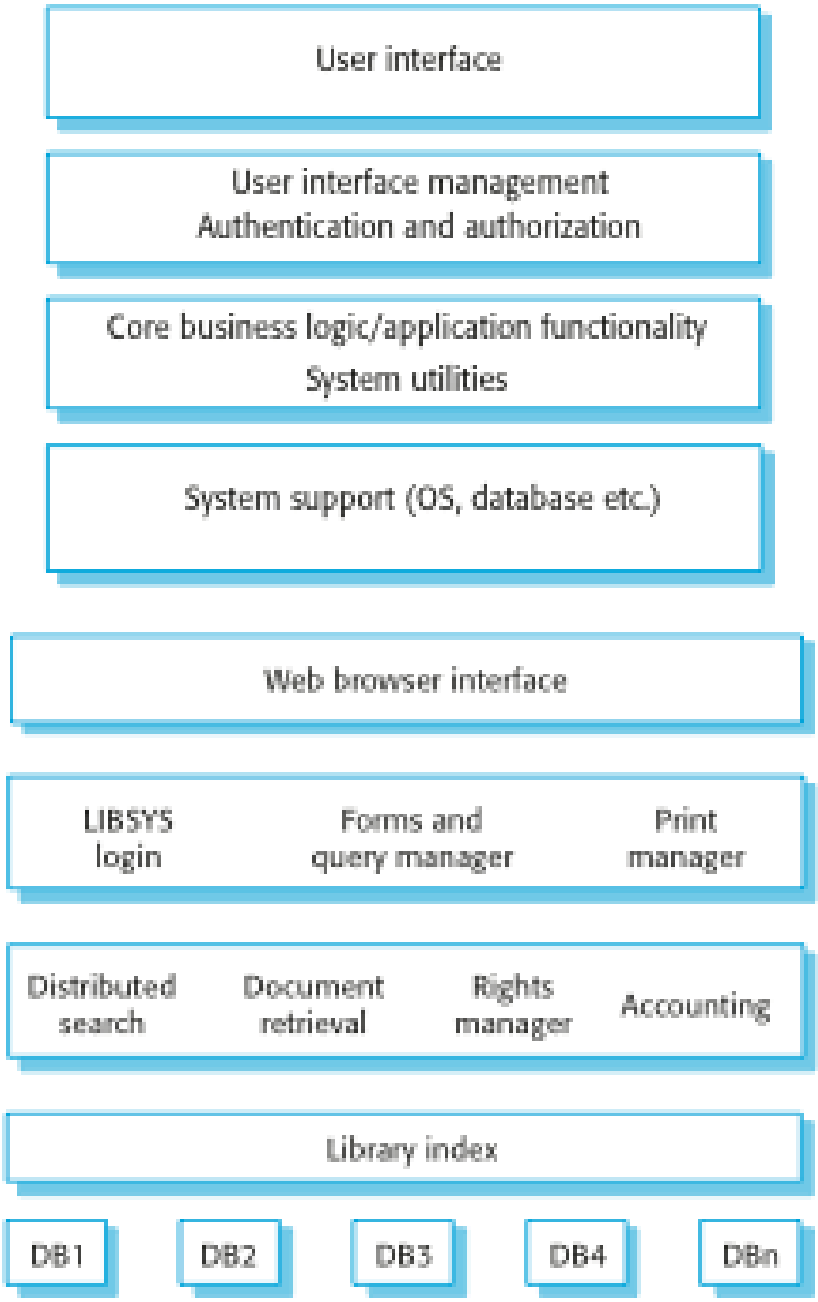
تصميم النظام على شكل طبقات في كل طبقة توجد الخدمات المتشابهة ويصبح التواصل بينها بالشكل التالي : كل طبقة تتواصل مع الطبقة التي تليها والتي تسبقها فقط مثل OSE

وغالباً الطبقات الداخلية هي التي تتعامل مع قاعدة البيانات ويكون الأداء سيء أما ال security يكون عالٍ.

إيجابياتها:

كلمت أردنا إضافة خدمات نضيف طبقات جديدة وهكذا فهي تدعم التطوير التزايدي.

سهولة الصيانة بسبب كثرة الطبقات.



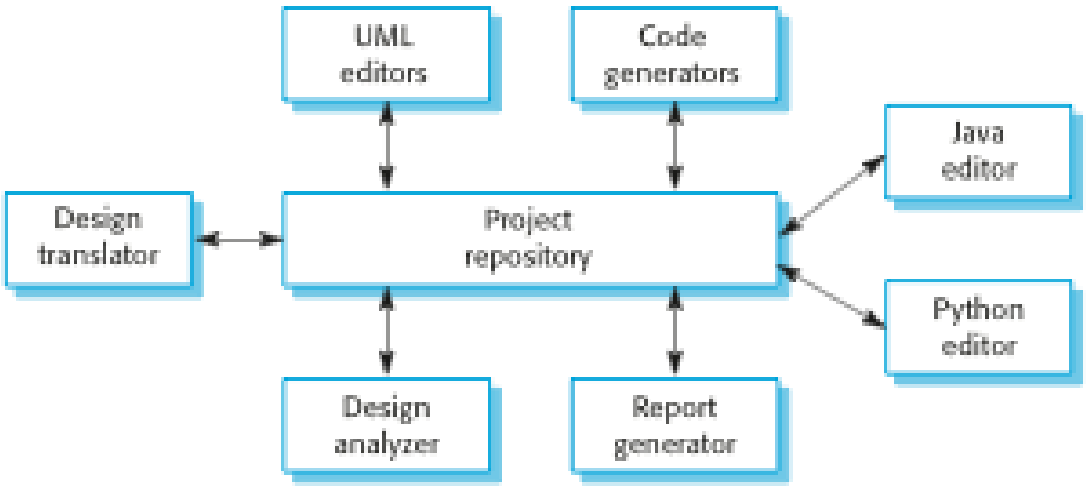
3. Repository Architecture :

تستخدم ال ERP: Enterprise Resource Planning وهي packages of software جاهزة تستخدم لإدارة المؤسسات والجامعات الضخمة ويوجد database أساسية كبيرة

أي لدينا أكثر من تطبيق تستخدم لنفس التابع مثل شركة لديها قاعدة بيانات مركزية وجميع الأقسام تستخدم هذه القاعدة.

الانظمة الجزئية تتبادل الداتا وتتشاركها وذلك عند وجود قاعدة بيانات ضخمة.

تستخدم أيضاً في IDE حيث نطبق عدة خدمات على نفس المعلومات مثل رسم المخططات وتوليد الكود..



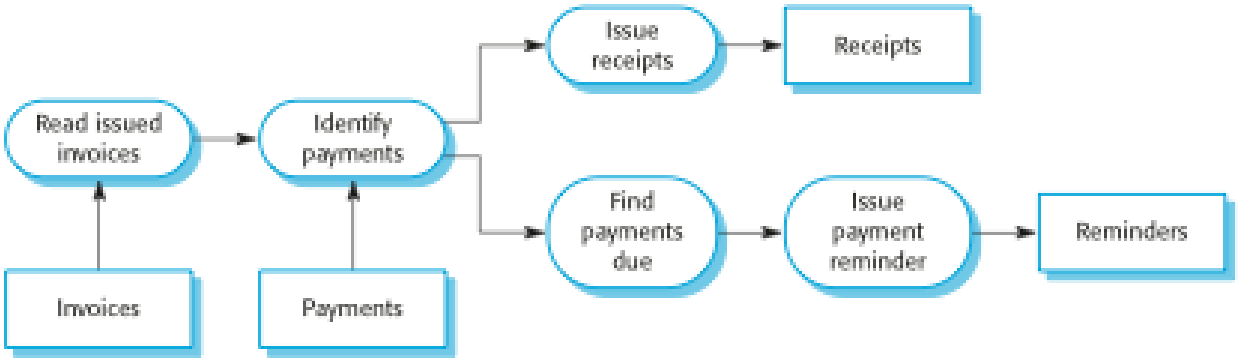
Client- Server Architecture.4

عندما لدينا مجموعة هائلة من الزبائن نريد تخديمهم بنفس الخدمات عبر **server** ولها أنواع :

N tiers – fat client –thin client

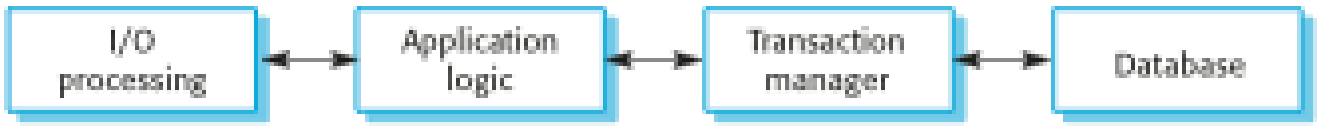
Pipe & filter Architecture .5

يستخدم في الأنظمة ذات العمليات التسلسلية في الامور الوظيفية مثل الكومبايلرات : خرج كل مرحلة دخل العملية التي تليها.



Application Architecture :

أول خطوة في عملية التصميم هي إيجاد المعمارية العامة للنظام قد تكون مناسبة لكل أنواع نظم المعلومات وهي المعمارية الطبقية وتتألف بطبقة خاصة بالتعامل مع الزبون وطبقة خاصة بالتعامل مع قاعدة البيانات وطبقة خاصة بمعالجة المعطيات



انتهت المحاضرة^^