Topic: 软件架构(Architecture)及设计

Sub topic: Architecture + CMMI 强化 Outsourcing 项目管理

By 高焕堂 2005/9/6

1. *何谓软件架构呢?*

IEEE-Std-1471-2000 *Recommended Practice for Architectural Description of Software-Intensive Systems.*

Achitecture -- The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [IEEE Std 1471-2000]
(Architecture 是一个系统的基本组织，它蕴含于系统的组件中、组件之间的相互关系中、组件与环境的相互关系中、以及呈现于其设计和演进的原则中。)

Architecture 就像高楼大厦的钢骨结构，将无数个 part 组合成为和谐的 whole。

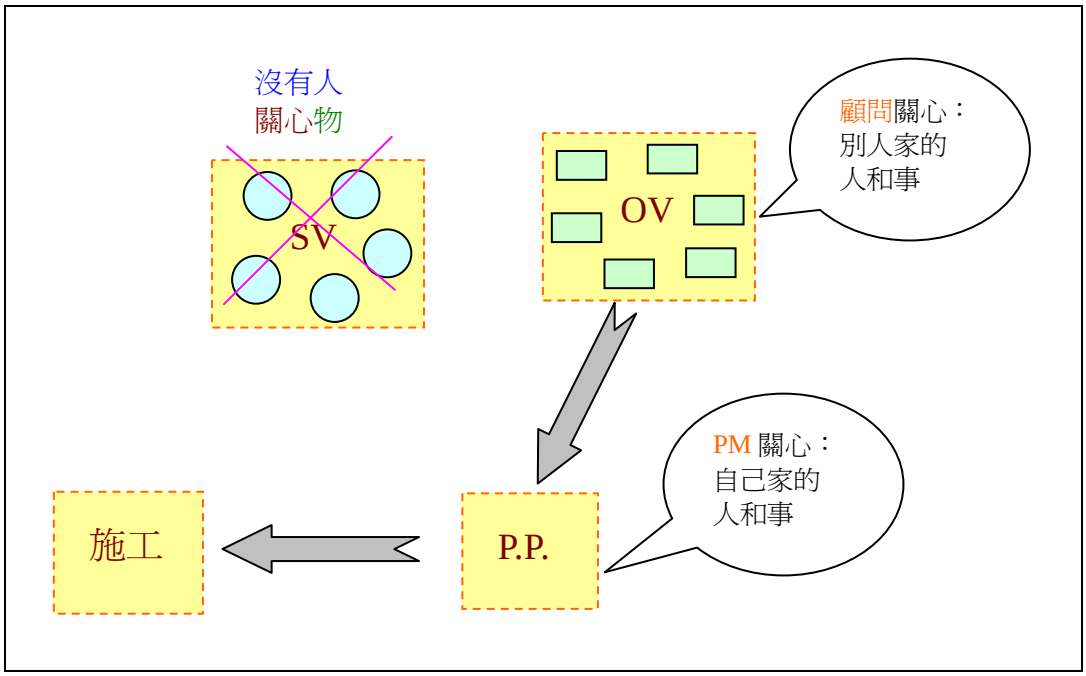
好的 architecture 能带来和谐、弹性、可靠的整体(whole)。同样地，好的系统 architecture 能带给企业和谐、弹性、可靠的整体信息系统。21 世纪软件架构师 Marc

Swewell 在其书 ----- *The Software Architect* 's *Profession* 里提到[Sew2001]：

> “Architectural design is a creative, sometimes mysterious, process that culminates in a plan for the construction of a structure, be it a building, a machine, a ship, or a software system or product. The design, *venustas*, is what unites the client, *utilitas*, with the finished structure, *firmitas*.”
> (架构设计是一个创意(有些神秘)过程，最终产出系统结构的实现计划，此系统结构可能是一栋建筑物、一部机器、一项软件系统或产品。藉由设计(*venustas*)把客户需求(*utilitas*)与最终建筑结构(*Firmitas*)融合为一体。)

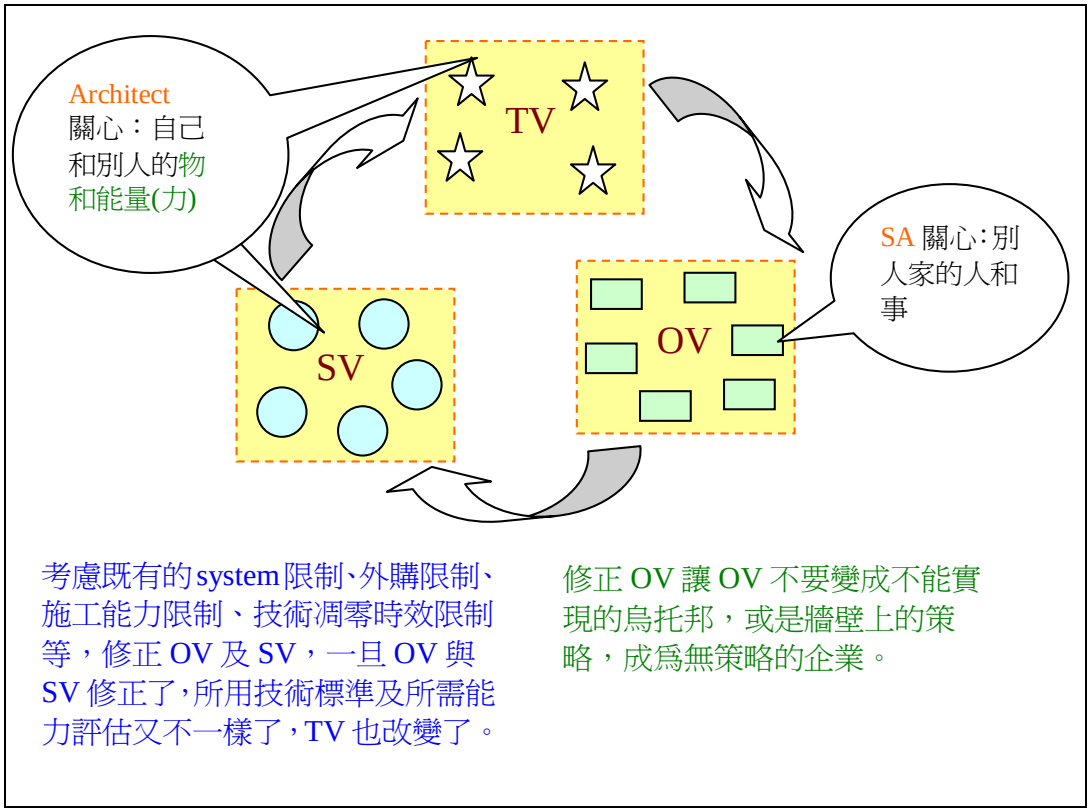
2. *架构设计与项目计划(P.P.)*

一般而言，客服顾问或系统分析人员(SA)只关心别人的人和事，也只关心如何 ”用” 物。而 PM 只关心自己团队的人和事。却没有人关心如何产出物，没有人关心物(system)的结构和组件之接口。如下图：

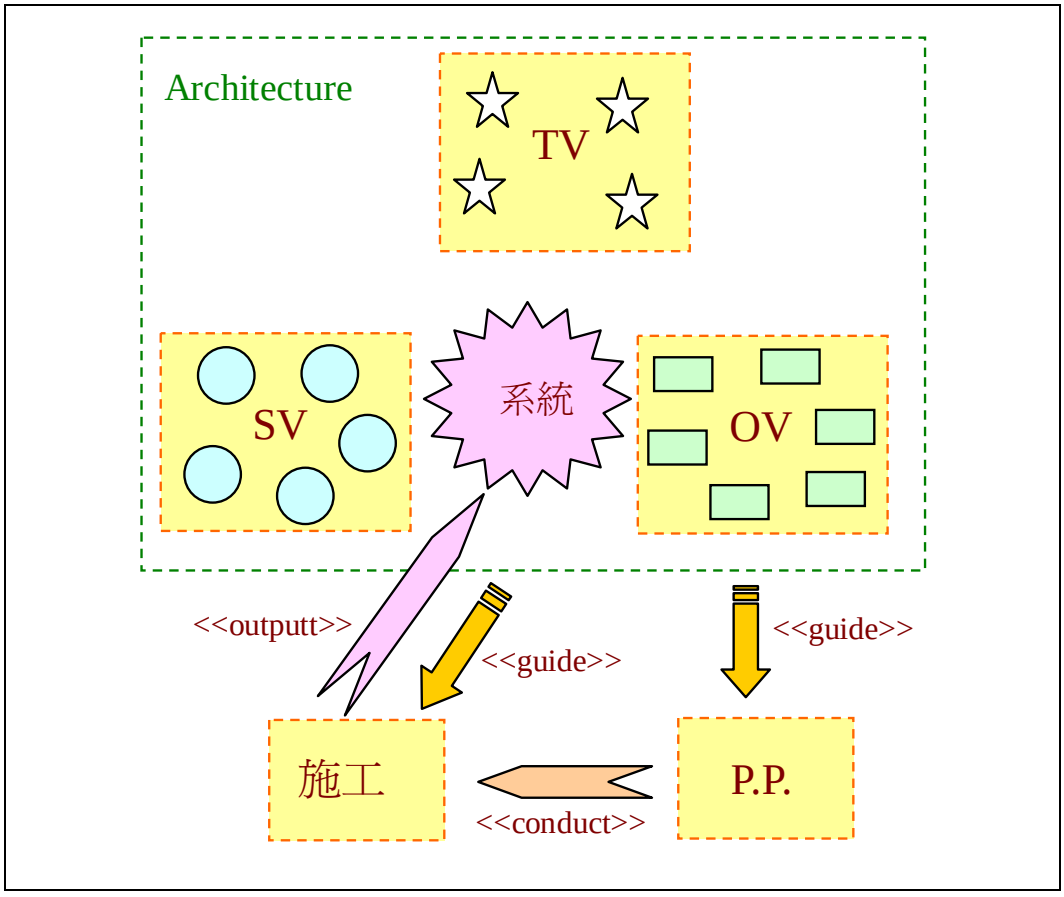


PS. OV ---- Operational View, SV ---- System View
TV ---- Technical View, P.P ---- Project Plan

所以，由 Architect 去 iteratively mapping vision (OV) to reality (SV & TV)，而修正 OV 及 SV 和 TV。三者 co-evolution 之后才会是和谐的 architecture。



唯有和谐的 architecture 才能导出最佳 how-to (right way to do)。



凡是 SA 所分析到的都不是核心 architecture，而都是核心 architecture 所必需 support 的。所以才称为需求。需求就是必 ”需” 要 ”求”architecture 加以支持的东西。但 architecture 不是从需求里找出来的东西。例如梁柱不是从屋顶找出来的，树干不是从树叶堆里找出来的东东。一个内部和谐的 architecture 并非就是真正和谐的 architecture，还要与其它同等级 architecture 互相融合(co-evolution)而得出和谐的上级 architecture，而且要保持长久和谐，才是真正和谐的 architecture。好的 SV 必须能与其它同等级的 architecture 的 SV 相连结而实现上层的 SV，来支持上层的 OV。

3. *架构设计与制程管理(CMMI)*

软件专家 Peter Merel 认为孔子的礼与软件开发管理(如 CMMI)有很微妙的相似处。Peter Merel 也提到[注 1]：

> “Traditional software methodology resembles a Confucian doctrine. It pursues perfection of process through the communication of stylized documents between technical specialists arranged in fixed roles.”
> (传统的软件开发方法，与孔子的礼非常相似。其追求完美的流程，规范软件人员都有所分、也有所归，各扮演固定角色，然后藉由制式的文档互相沟通，形成一个严谨的工作流程。)

孔子经由其修练法门，达到「约」之以礼，呈现出美好的秩序和规律。论语 为政篇里 孔子有曰：

「吾十有五而志于学，三十而立，
四十而不惑，五十而知天命，
六十而耳顺，
七十而从心所欲不逾矩。」

像 CMMI 也经由类似的修练，分为 5 个修练的层级(level)，如下：

第 1 级谓之 **Chaos** (混沌无序)
---- 相当于：不学无礼
第 2 级谓之 **Repeatable** (重复练习)
---- 相当于：十五志于学，学而时习之
第 3 级谓之 **Defined** (确立程序)
---- 相当于： 三十而立
第 4 级谓之 **Managed** (量化可预测)
---- 相当于：四十而不惑
第 5 级谓之 **Optimal** (止于至善)
---- 相当于：五十而知天命，六十而耳顺，七十而从心所欲不逾矩。

礼和 CMMI 都不是最高境界，如果能参悟礼和 CMMI 的根源，将更能发挥其真谛和效益。在《礼记・孔子闲居》里，

> 孔子曰：「夫民之父母乎！必达于礼乐之原，以致五至， 而行三无，
> 以横于天下，四方有败，必先知之。此之谓民之父母矣。」
> (白话: 所谓民之父母，必定能通达于礼乐的本源，达到「五至」并实行「三无」于天下，当四方有危难之前能预先知道并化解危机，如此就可称为民之父母了。)

孔子指出礼的更高境界在五至，我们可以藉由孔子的高度智慧而探索礼和 CMMI 的更高境界。数年前才被发现的简帛 ---《上海博物馆藏战国楚竹书（二）·民之父母》记载孔子的「五至」是：

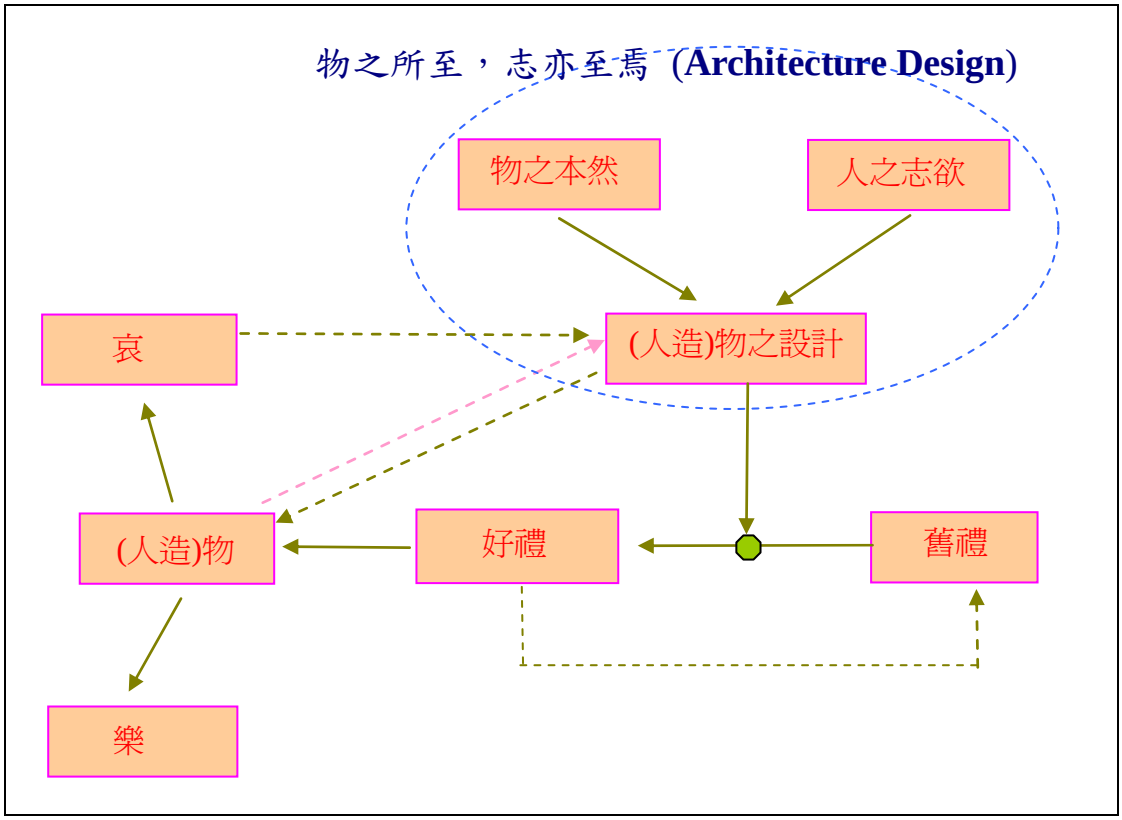
> 「物之所至，志亦至焉。志之所至，礼亦至焉。
> 礼之所至，乐亦至焉。乐之所至，哀亦至焉。
> 哀乐相生，君子以正。」

笔者的解释如下：

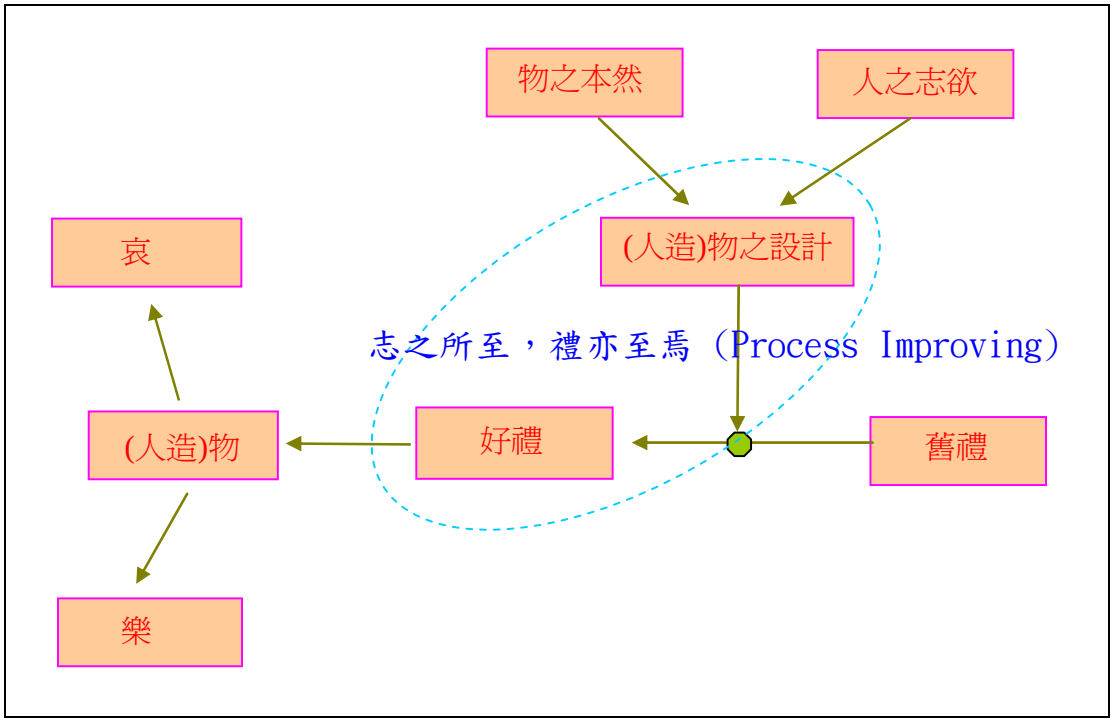
物 --- 就是软件系统(System)及其架构(Architecture)
志 --- 就是人们(包括 User 和 Developer)的需求(Requirements)
礼 --- 就是制程管理制度(Process Management)如 CMMI
乐 --- 就是需求获得满足
哀 --- 就是需求无法满足

兹进一步解释如下：

物之所至，志亦至焉。
--- 系统(物)回归其本然,让众人的需求(志)和谐而不冲突。
志之所至，礼亦至焉。
--- 基于和谐之需求下就能制订美好的项目执行计划(礼)。
礼之所至，乐亦至焉。
--- 依据计划开发出合乎自然的系统满足众人的需求，不亦乐乎(乐)。
乐之所至，哀亦至焉。
--- 环境会变迁，人们需求改变了，系统无法完全满足众人的需求，抱怨于焉而生(哀)。
哀乐相生，君子以正。
--- 系统重新回归本然，众人之需求又获得和谐,调整项目计划及系统，众人又满足了其乐融融(乐)，如此哀乐相生循环不已，有智慧者都知道这是常态。



架构设计图(architecture design artifact)是「物之所至，志亦至焉」结果的知识性表现，可提供给管理者(PM)作为订定或修正礼(process)的依据，而获得好的礼。于是，达到「志之所至，礼亦至焉」了，如下图所示：



4. *Outsourcing Management(委外项目管理)*

由于软件业与硬件业之间有个显著的不同点，就是：软件业的业主(A 段)与承包商(B 段)两段之间需要密切的语言沟通。文化背景差异又会影响语言沟通的效果。这也是为什么美国发展海外委外(Offshore Outsourcing)时，会优先选择印度的主要因素。而欧洲发展近海委外(Near-shore Outsourcing)时，会优先选择东欧诸国。

比较成熟的计算机硬件业里，欧美国家提供明确的架构设计，所有组件的接口都一清二楚，走向 Architecture-based 的委外模式，所以台湾新竹与美国硅谷之间分工明确。较不成熟的软件业里，欧美国家只提供系统使用需求(user interface)，未提供组件接口(component interface)，走向 Requirement-based 的委外模式，因为「物」之结构并不够清晰明确，所以印度必须派遣大批人员远赴美国，并依赖语言和文字的密切沟通。由于语言的限制，台湾软件业不容易像硬件业一样争取巨大的委外代工，而印度有语言之优势，是台湾软件业不容易与印度竞争的主因之一。

例如， Arora1, Arunachalam, Asundi, Fernandes, *The Indian Software Industry*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=198968

在其调查报告中指出，依据统计，42.7%的工作在海外进行。这意味着在美国业主端的工作占了一大半，并非大部分在印度做的。如果把在美国做的部分称为 A 段，而印度做的部份称为 B 段。则 A 段占 57.3%，而 B 段占 42.7%。B 段依靠 CMMI 来强化其制程管理，A 段则靠系统 Architecture 设计来强化。产品 Architecture 设计与 CMMI 制程管理在异地互相辉映是成功的重要模式。例如：

Kesav Nori, Deepak Kumar, *Building Change-resilient Software*, http://www.softwaredioxide.com/channels/PersonView.asp?id=6351

文章里提到，印度 Tata 公司的 CIO ---- Kesav Nori 提出：

> "Architecture-based development is reckoned to be one of the best practices of software development. However, the real and the hidden meaning of software architecture is understood by few professionals."
> (以架构为基础的开发是软件开发的最佳实务之一。然而只有少数人了解软件架构的真谛和内涵。)

再如 Vinod Varma, "Moving software engineering from hype to reality: a long journey" 文章里提到：

"My survey of software engineering literature around the world reassures me that I am not alone in this observation and neither is this phenomenon restricted to India alone. An architecture which is resilient to changes, encapsulating the core logic & rules, easy to adapt etc could be said to hold the key. Identifying patterns and templates for reuse with the established optimized solutions from within the organization, identification of variation points and the use of in-house, off-the-shelf, or hybrid solutions for automation could greatly

enhance the productivity.”

当软件业愈来愈成熟时，就愈重视架构设计，则美国/印度的软件 Outsourcing 模式就愈来愈走向美国硅谷/台湾新竹的硬件分工模式。A 段就不太需要藉由 CMMI 去干涉 B 段企业的人员修练与管理制度了。更重要的是，对于英语语言的依赖就愈来愈少了。由此观之，目前 Architecture(如 SOA)新潮流对大中华地区软件业争取 Outsourcing 业绩会有极大帮助，大幅提升国际竞争力与印度互别苗头。

5. *Architect 的职责*

柳宗元的「梓人传」里强调梓人(Architect)的角色，在于：产品设计与分工生产规划与领导。产品设计的内涵如下：

曰：「吾善度材，视栋宇之制，高深方圆短长之宜…。」

这是产品的整体架构规划和组件的接口设计，确保产品的分工整合。至于分工生产规划的内涵如下：

曰：「… 吾指使而群工役焉。」

这是人员的领导，确保团队和谐一致。Architect 的价值和重要性如下:

曰：「… 舍我，众莫能就一宇。故食于官府，吾受禄三倍；
作于私家，吾收其直太半焉。」

Architect 的修练流程与 CMMI 修练不一样，是互补的。梓人传里说明梓人的重要特质下:

「彼将舍其手艺，专其心智，而能知体要者欤！」

这弥补了传统 CMMI 的不足，传统 CMMI 强调手艺的精准无误，并且要能量化。梓人传里也有更清晰的说明：

「不衒能，不矜名；不亲小劳，不侵众官；….. 犹梓人之善运众工而不伐艺也。」

其中，「善运众工」(Outsource)是最重要的修练。传统 CMMI 的修练，并不适合 Architect。如果把传统 CMMI 的修练误用于 Architect，将适得其反，后果将不堪想象，就像梓人传里的说明：

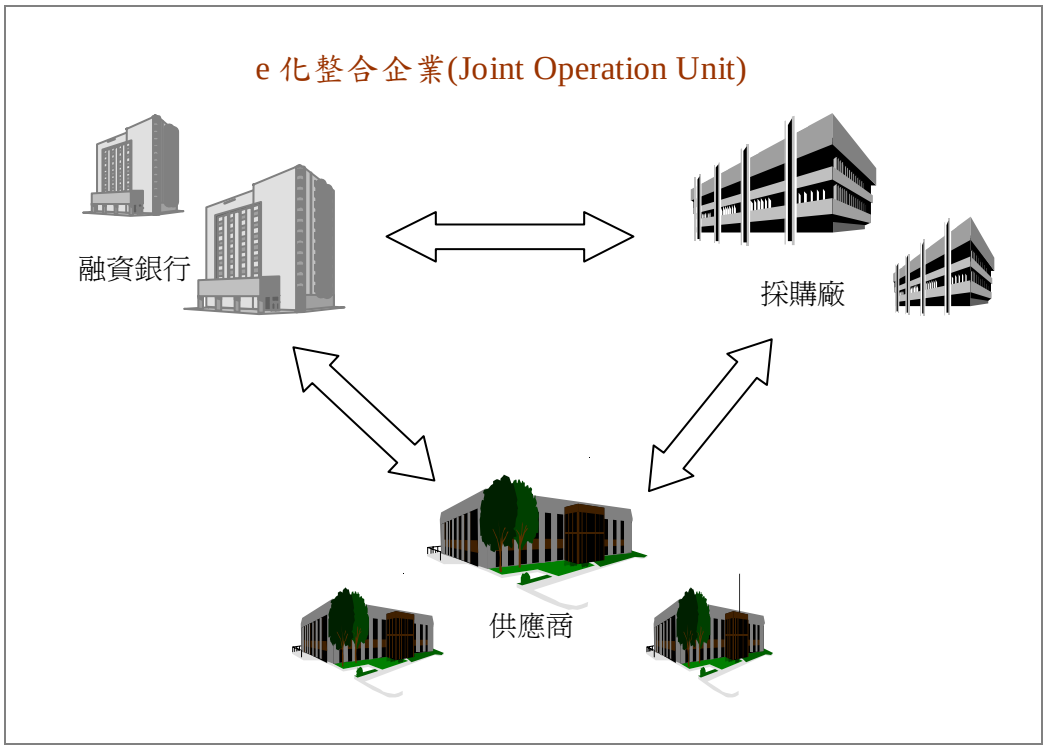
「犹梓人而不知绳墨之曲直，规矩之方圆，寻引之短长，姑夺众工之斧斤刀锯以佐其艺，又不能备其工，以至败绩用而无所成也！」

果真如此，则 Architect 会见树不见林，无法做好其整体架构设计及分工生产规划。

6. *SOA 架构设计之案例*

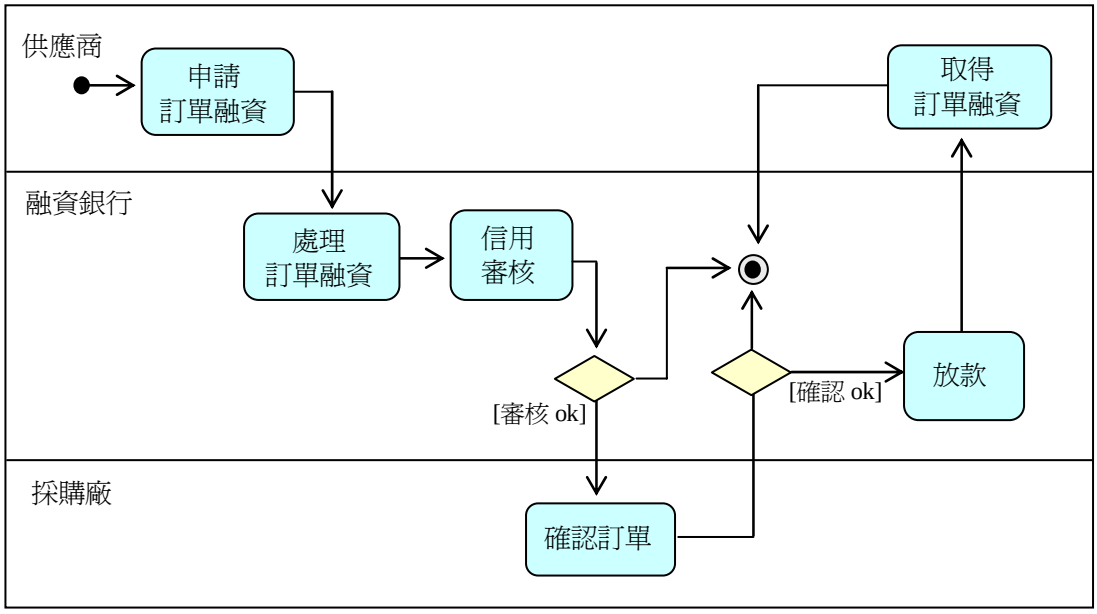
----- *采取WSF(Web Service Façade)*

兹举跨企业『订单融资』系统整合为例，说明如何以软件主机板(即 WSF)之概念而擘划订单融资系统整合架构。订单融资的商业情境是：采购厂(如大同公司)向供货商(如大铭精密公司)发出购买马达的订单，供货商就立即携带订单向银行(如中国信托)办理融资贷款，银行向采购厂确认订单之后就放款给供货商，供货商以该款项去购买材料来生产马达，完成之后，运送到采购厂，验收之后采购商才付款给银行。如下图：



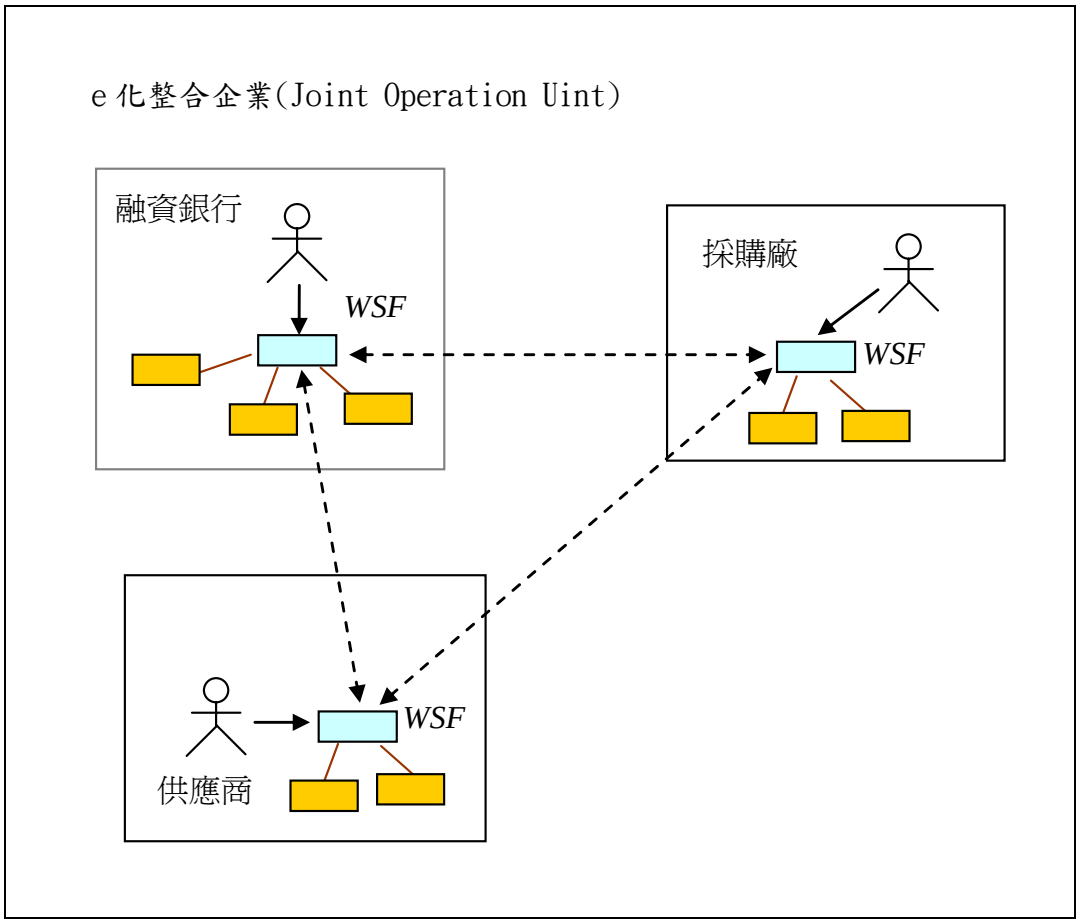
(订单融资整合服务)

这跨企业的订单融资之整合业务流程如下图：



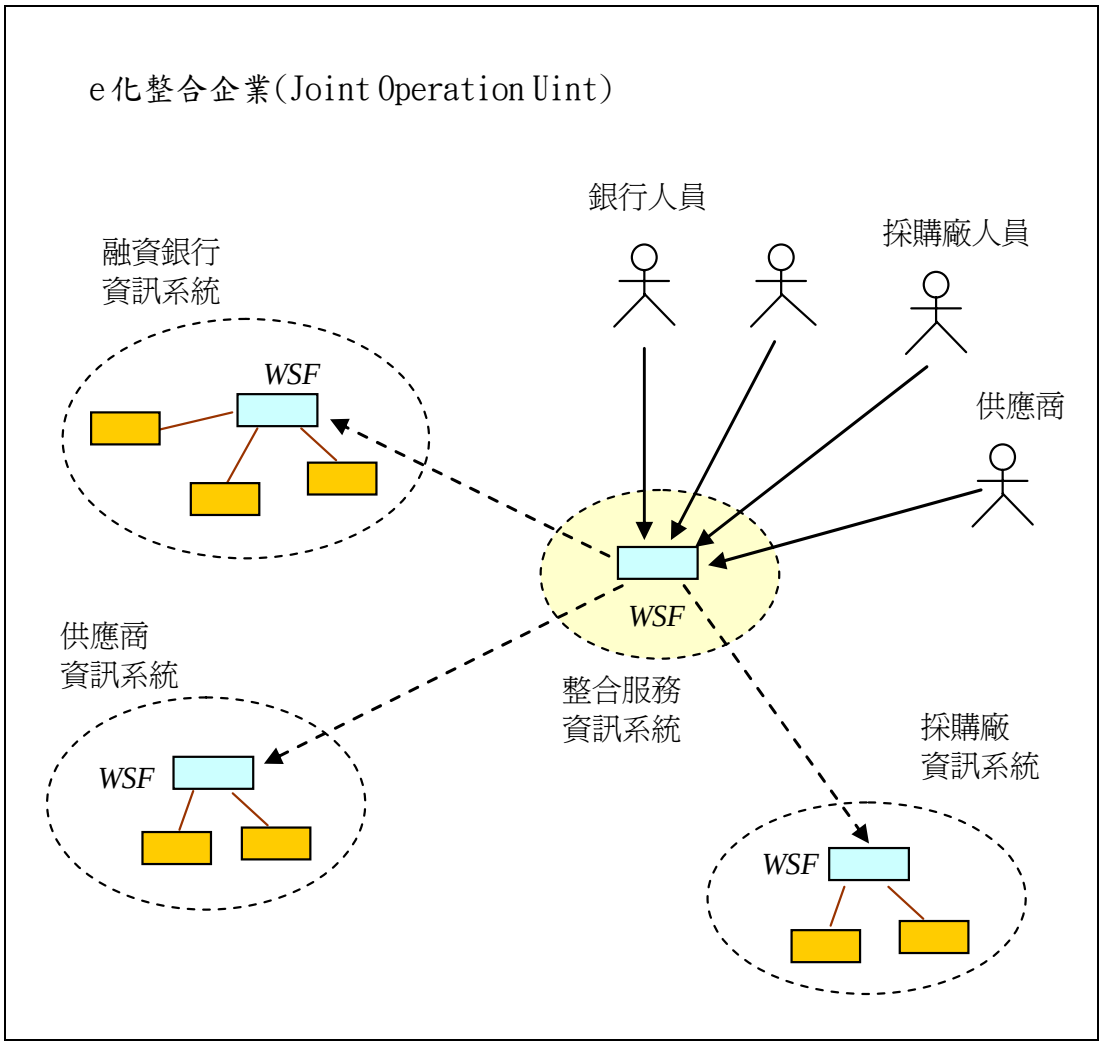
(订单融资服务基本流程)

想象在各个企业里，各规划一个 WSF 模块，作为该企业的软件主机板(Software MB)，一方面整合既有系统之服务，一方面作为对外沟通之窗口。这三个 WSF 具有一致的接口能顺利地互相沟通，如下图：



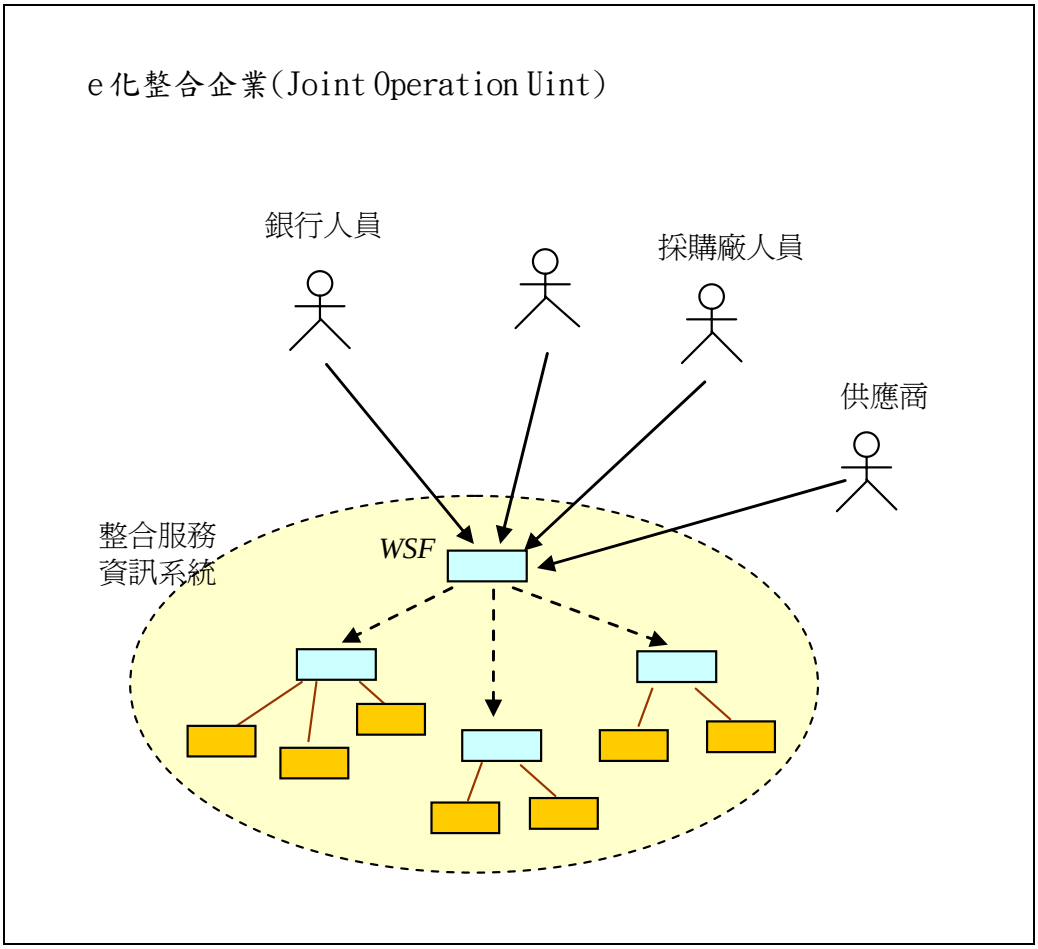
(Software MB 整合架构之一)

也可以进一步规画一个上层的 WSF 来整合这三个小 WSF，让全世界各地的使用者，包括客户、各企业的业务人员等，都使用上层 WSF 的服务，如下图：



(Software MB 整合架构之二)

假如有一天，这三个企业合并成为一个大企业，此时可以将各地的信息系统集中到一个地方，以便于管理和服务，如下图：



(Software MB 整合架构之三)

Software MB 整合架构的形式可以千变万化的，想一想，计算机网络整合了全球各地的硬件主机板，而且形式无限变化。藉由硬件主机板的对比，的确能促进对 WSF 的体会，激发更多形式之整合架构设计。

[**Sew2001**] Marc Sewell, Laura Sewell, Software Architect's Profession: An Introduction, Prentice Hall, 2001.