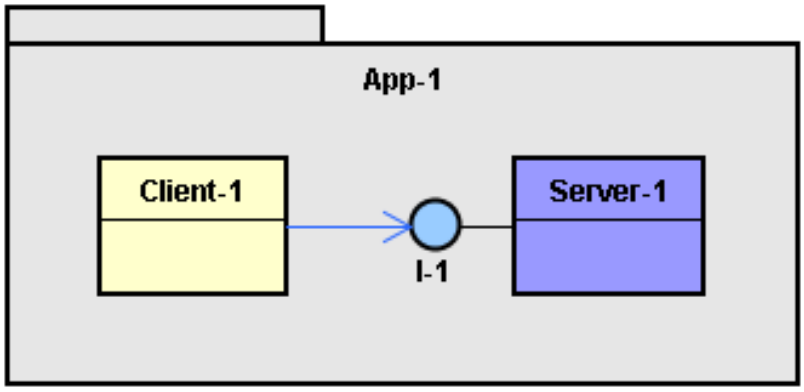
G06_接口设计之美_神奇的通用性接口

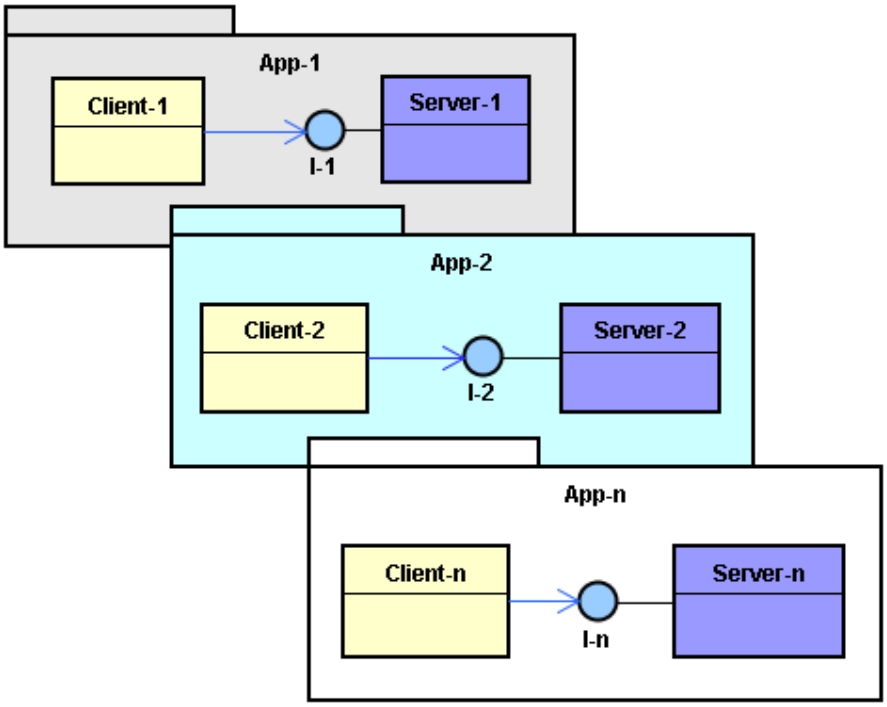
内容：

1. 通用性接口的涵意

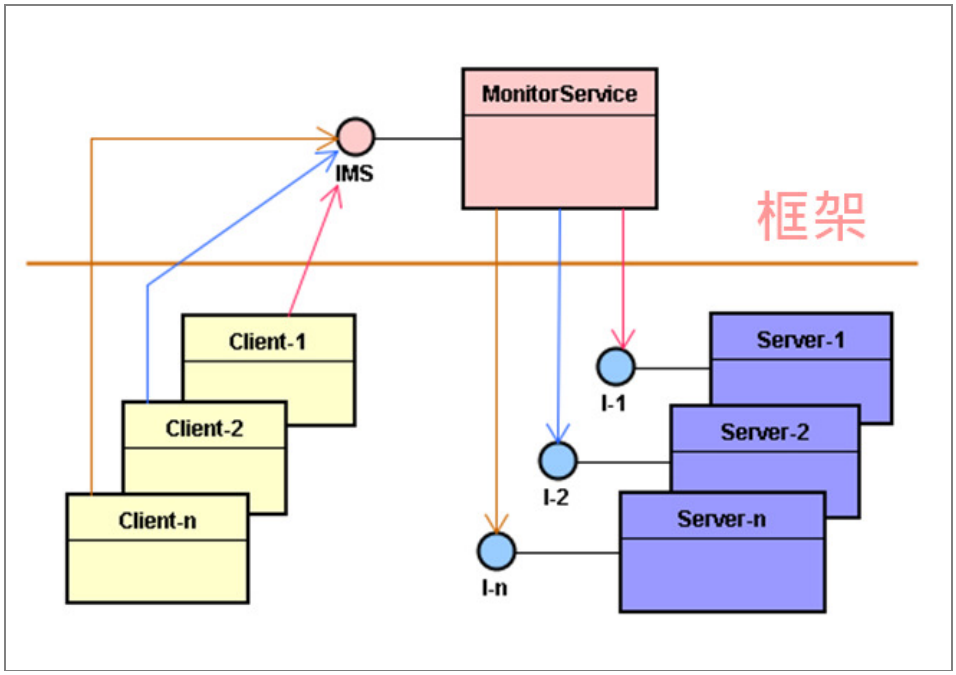
“通用性”与”特殊性”其实只是相对性而已。在下图里的<I-1>接口，可明显看出它是专属于<Client-1, Server-1>的特殊性接口。



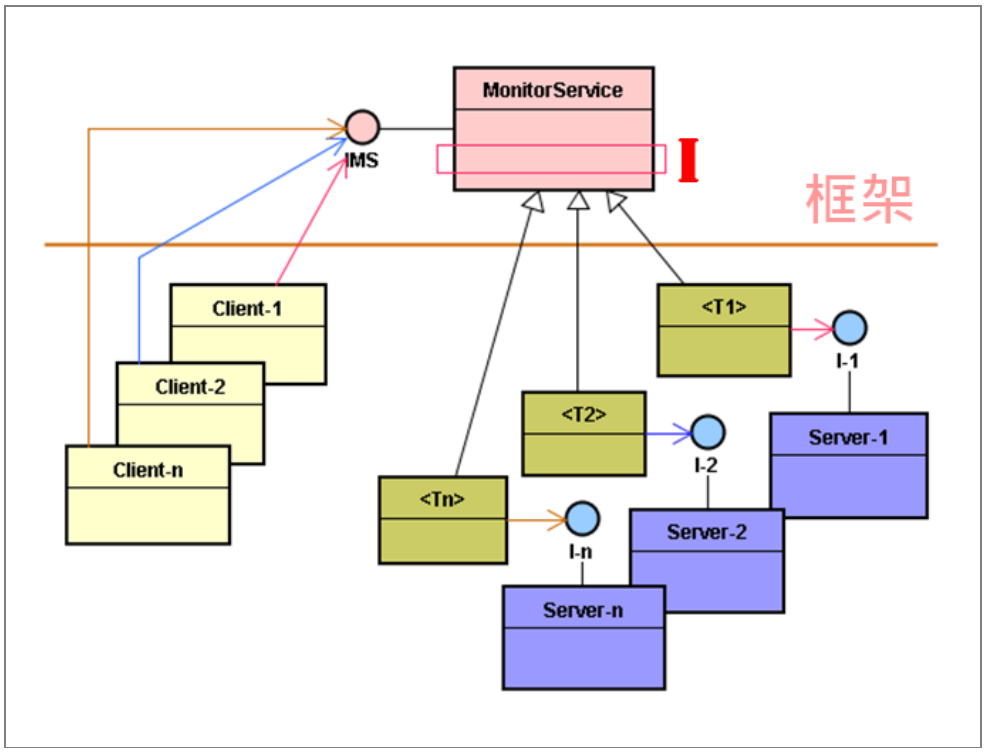
随着企业业务需求的增加，软件系统会含有愈来愈多的特殊性接口；如下图里的<I-1>、<I-2>和<I-n>等。



为了整合这些众多的特殊性接口，通常会设计出通用性接口，来试图统一他们，进而监控(Monitor)它们。如下图里的 IMS 就是一个通用性接口了。



其实，通用性接口常常示成双成对的，例如上图里的<IMS>是对 Client 端的通用性接口；此外，还常常设计一个对 Server 端的通用性接口，它就是大家熟悉的 EIT 造形里的<I>了。如下图所示。



上述的架构，看来颇为合理而美好。那么，又如何设计这两个通用性接口的内涵呢？那么，又如何从通用性的<I>转换成为<I-1>、<I-2>等特殊性接口呢？这就是本单元课程的焦点了。☆

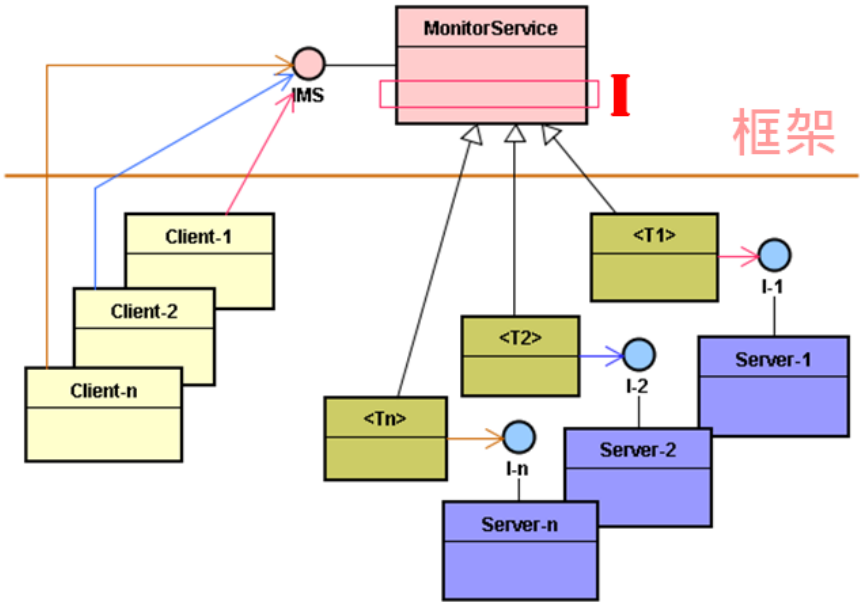
G06_接口设计之美_神奇的通用性接口

内容：

◇ 基本技巧：通用性<I>设计方法

前言：

- 上一节说明了，通用性接口常常示成双成对的，例如下图里的<IMS>是对 Client 端的通用性接口；此外，还常常设计一个对 Server 端的通用性接口，例如下图里的<I>了。



- 接下来，就先针对<I>来探讨其设计思维和方法。

2. 通用性<I>创造重构(Re-Factory)的自由度

架构设计的两项关键议题是：

- 架构设计如何迅速落实为代码。
- 架构师团队如何给自己创造重构的自由度，以及支持开发者重构的空间。

2.1 迅速落实为代码

架构师以 EIT 代码造形表述设计；让开发者直接对应到代码。代码造形就如同专块，建筑师叙述如何以砖块组合出形形色色的建筑物；施工者就烧出专块，

并按步就班组合起来。其中，EIT 造形成为架构思考的简单元素，然后从简单中组合出复杂架构，而框架则是产出的代码层级的架构(亦即，计算机可执行架构)。

基于 EIT 造形，架构师和开发者都能从简单组合出复杂。亦即：造形很简单，内涵可复杂，重复地组合。让用户获得从简单中叫出复杂的满足感。亦即：优质的用户体验。

2.2 重构的自由度

架构师团队如何给自己创造重构的自由度，以及支持开发者重构的空间，是架构设计的关键议题。这种自由度，决定于架构师是否能仔细分辨出：关注<未来的决策>与关注<今天决策的未来性>的微妙差异了。愈是能关注<今天决策的未来性>，而不是关注<未来的决策>，就愈能创造未来重构的自由度。

例如，EIT 造形和框架的主角都是接口<I>，愈是关注<目前决策的未来性>时，就愈会想去设计通用性(General)<E>和<I>来包容未来<T>的多变化。而一群<E&I>的巧妙组合，就成为框架了。

由于 EIT 造形具有重复组合的特性，人们可以组合出多层级 EIT 造形体系的结构，进而设计出多层级的框架，就能创造更大的重构自由度。例如，上层 EIT 造形的<I>能包容用户需求<T>的未来变化；而底层框架则能包容系统平台特殊模块<T>的未来变化。用户需求与平台模块之间藉由两层 EIT 造形的通用性<I>来衔接与组合，而创造了弹性的重构空间。

3. 基本技巧：通用性<I>设计方法

3.1 基本概念

刚才说过了，架构设计包含两个层面： 1)思考设计； 2)表述设计。其中，架构师最关键的职责是接口<I>的设计和表述(Represent)了，也就是包含两个层面：1)如何进行设计接口<I>； 2)如何清晰表述接口<I>。

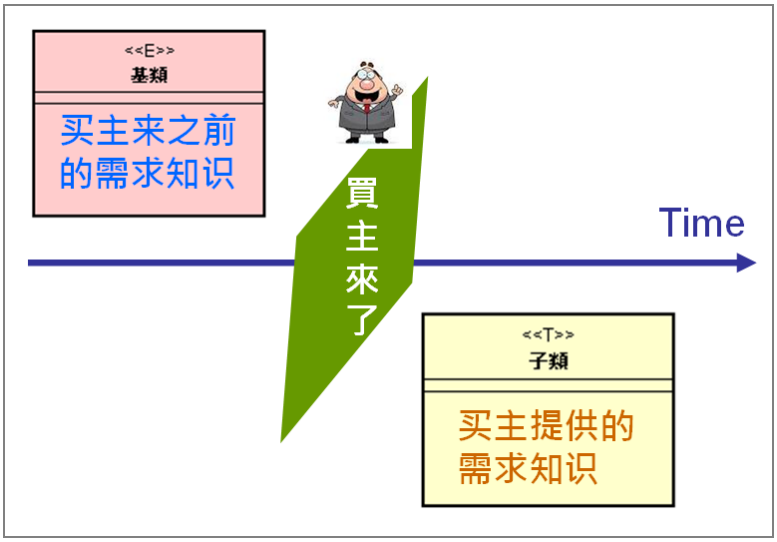
之前，我们大部分介紹如何將 EIT 造形應用於架构的表述上；亦即，架构师藉 EIT 造形来清晰而明确地传达接口的设计(Design)与定义(Definition)给开发者，让开发者能基于架构而顺利展开后续的详细设计，并迅速落实为代码，进而测试与回馈来驱动敏捷跌代过程。

其实，EIT 造形除了用来”表述设计”之外，也很适合于”思考设计”层面上。尤其对初级架构师而言，依循 EIT 造形的引导，能够找到潜藏不明的<I>，

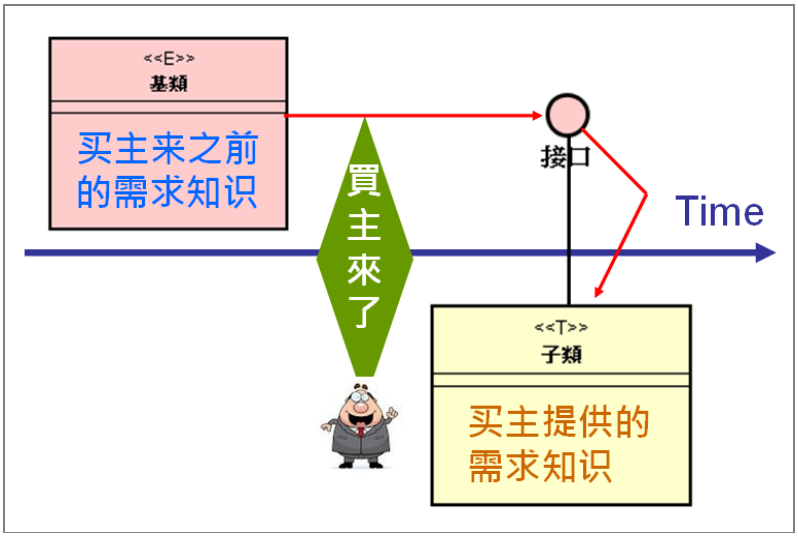
或者能激发创意，无中生有地创造了新的<I>。例如，初级架构师可藉由 EIT 造形来思考和实践<目前决策的未来性>，创造重构的自由度。

3.2 基本技巧：演練<目前决策的未来性>

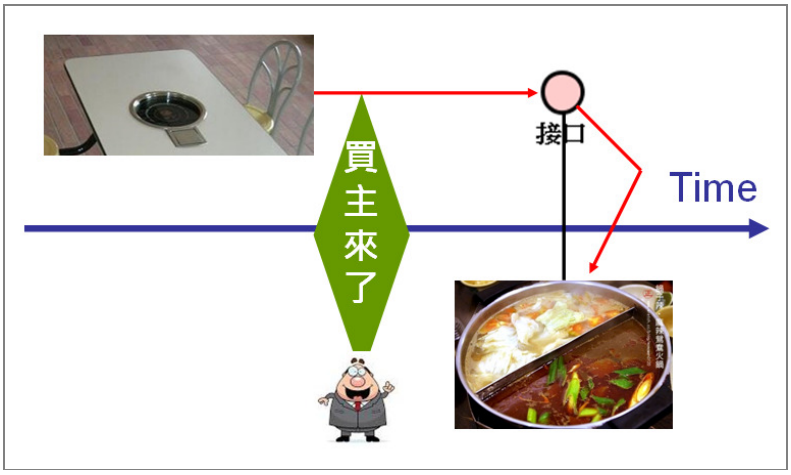
EIT 造形的<I>很适合做为通用性接口的起点(Simple Design)。<I>就是将<E>与<T>分离之后的整合点。<E>与<T>的分离，就是”架构师知识”与”买主知识”的分离。



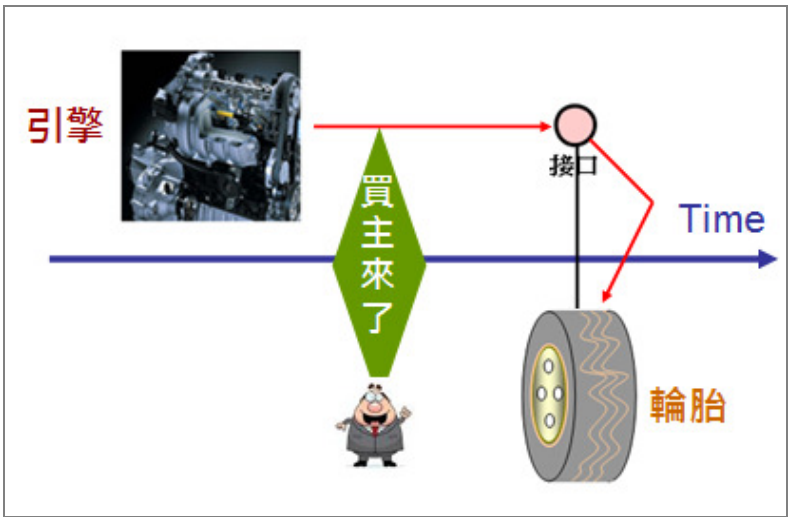
架构师知识与买主知识的获取，有时间落差，常依据”买主来到”做为时间切割点。



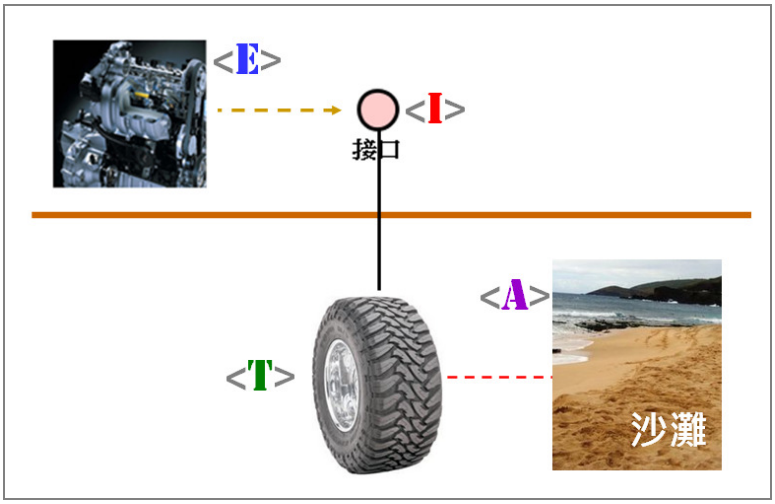
例如：



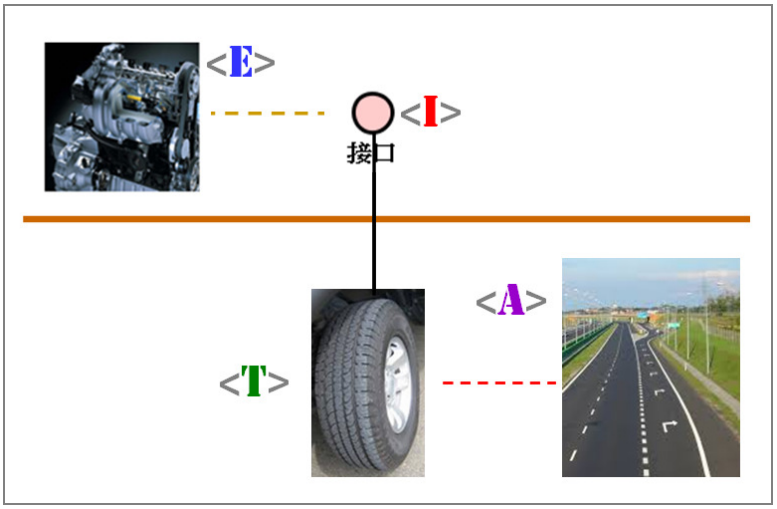
再如：



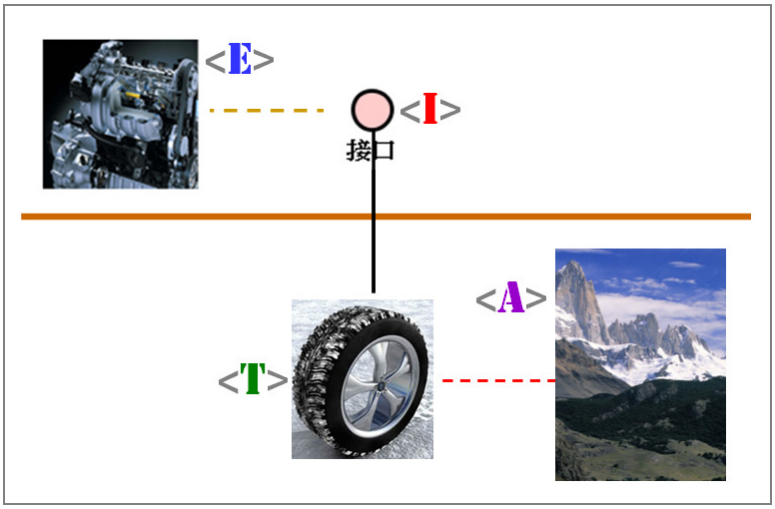
典型的 EIT 造形，<E&I>是一起设计的，<I>的定义权属于<E>设计团队的。从<E>的视角来看，此<I>是<E>所专属的。从<T>的视角来看，此<I>是各<T>所共享的；<E>也是各<T>所共享的；亦即两者都是通用性的。<E&I>表达了各买主(或客户)的通用性功能；而<T>则表达了各买主的特殊性需求。通用性的<I>，有两层意义：1) 容纳买主需求(或选择)的未来变化，或容纳新买主的新选择；2) 限制买主的选择范围。例如，买主买了车子之后，未来随时可以改变选择(沙滩、公路或高山)；展现出设计师目前决策的未来性。例如，买主未来决定将车子要到沙滩上跑时，就更换新轮胎，如下图：



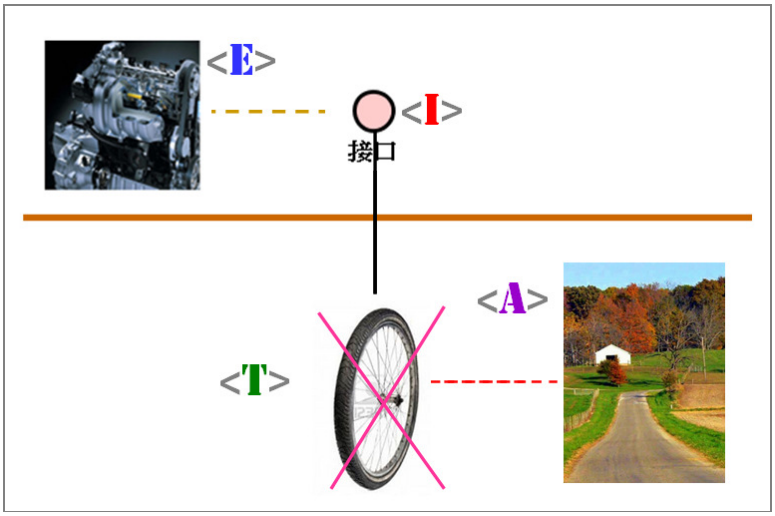
再如，买主未来又改變需求，决定将车子開到高速公路上时，就更换新轮胎，如下图：



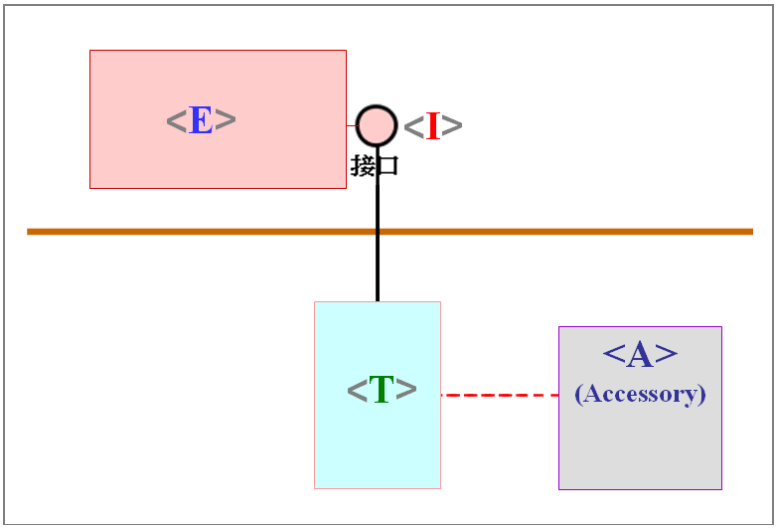
再如，买主未来又改變需求，决定将车子開到高山雪地时，又可隨時换新轮胎，如下图：



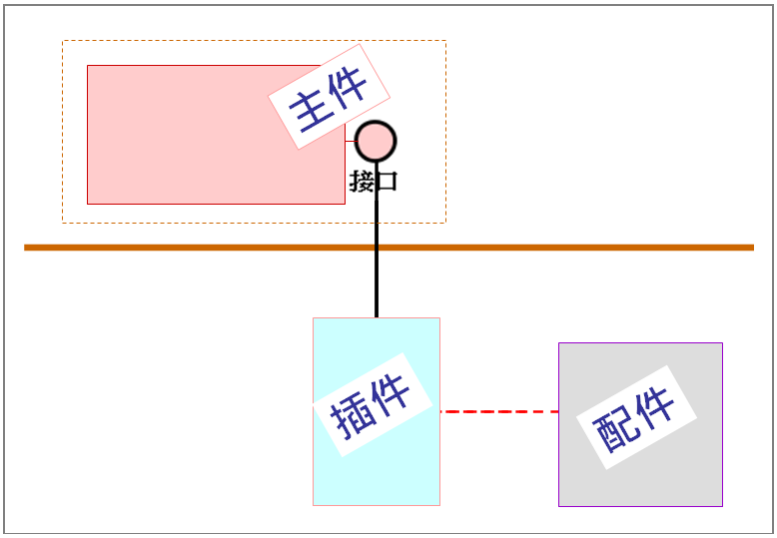
此外，EIT 造形也限制了买主的选择范围(藉由<I>来限制<T>和<A>)。例如，EIT 造形會清晰表述何種<T>才被允許裝配上來。例如，下圖就不是被允許的了。



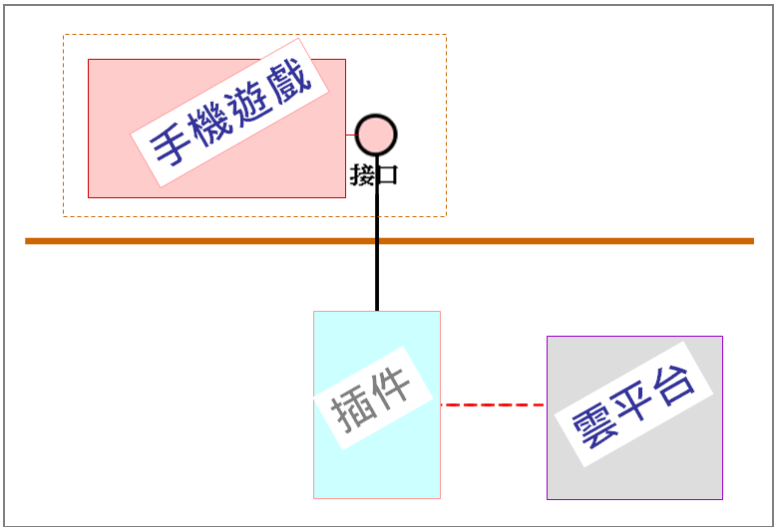
其实，EIT 造形只清晰地表述出很通俗的<主件、插件、配件>的组合关系而已。我们很容易将上述的汽车 EIT 造形，对映到软硬件的 EIT 造形，如下图：



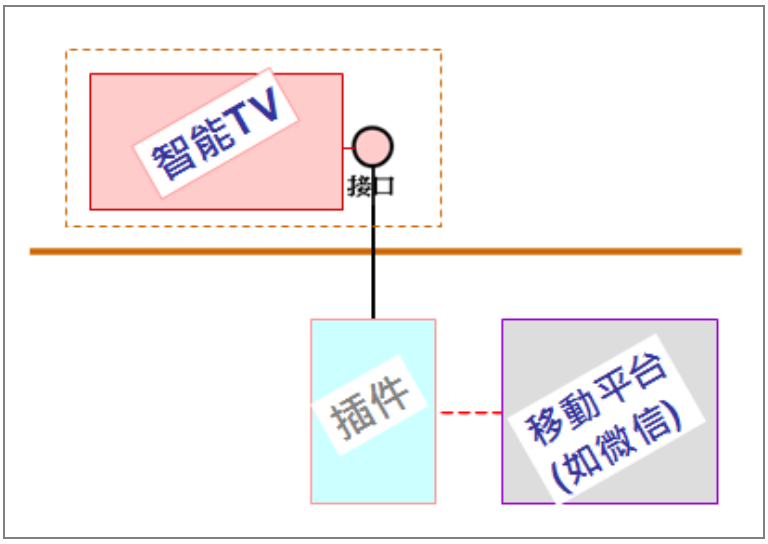
这图里的<E>、<I>、<T>、<A>共 4 个要素，就对映到通俗的<主件、接口、插件、配件>的组合关系而已。如下图：



兹将上述 EIT 造形应用于游戏软件架构上，手机游戏框架提供<E&I>，可以让买主(或用户)未来选择各种搭配的云平台。当买主未来又改变需求，决定使用新的平台时，只要随之更换新的<插件>即可了。如下图：



茲将上述 EIT 造形应用于智能电视(TV)平台上，智能 TV 提供<E&I>，可以让买主(或用户)未来选择各种搭配的 OTT 平台(如选择微信或微博等)，就能将家庭里的信息，透过智能 TV 而推送到微信、微博或 Skype 等 OTT 平台的客户端屏幕上。当买主未来又改变需求，决定使用新的 OTT 平台时，只要随之更换新的<插件>即可了。如下图：



买主在未来时间里，可能会改变他对配件<A>的选择(即改变决策)；每一项选择都是该买主的特殊性需求。于是，买主委托 App 开发者改写<T>代码来表达<A>的特殊性，并符合通用性接口<I>，来与<E>重新组合起来。

未来新的买主来了，让他在<I>的限定范围里进行他对配件<A>的选择(做决策)；其选择都是该买主的特殊性需求。于是，买主委托 App 开发者改写<T>代码来表达<A>的特殊性，并符合通用性接口<I>，来与<E>组合起来。

架构师设计了通用性接口<I>来容纳买主未来决策的改变，这也意味着：设计师的目前决策(决定如何定义接口)具有高度的未来性。 EIT 造形让架构师、开发者都具有整体观，间具通用性和特殊性的考虑；因而让整体系统具有高度的未来性和敏捷性。☆

G06_接口设计之美_神奇的通用性接口

内容：

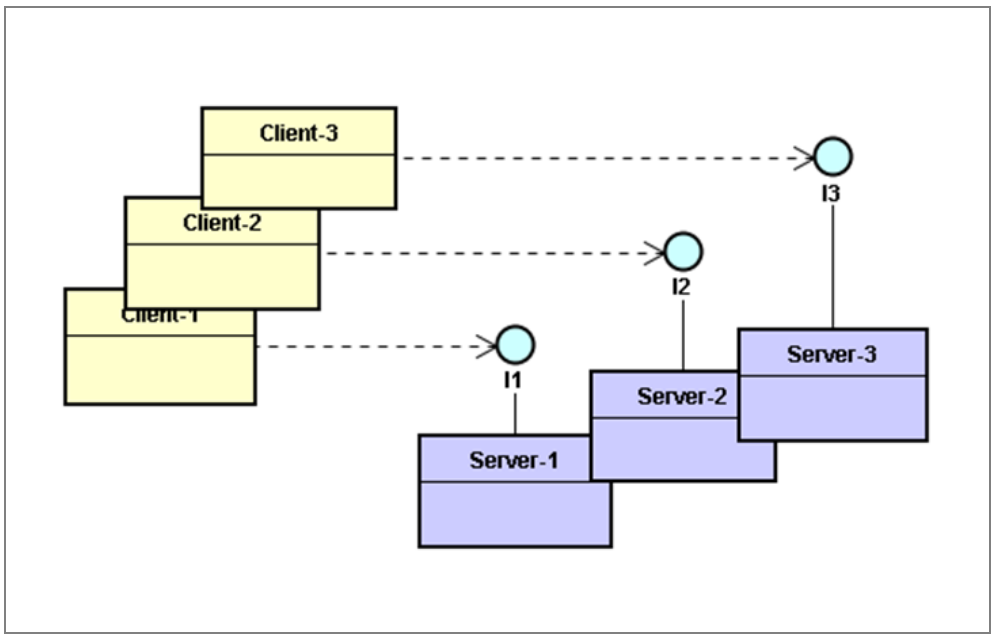
◇ 对 Client 端的通用性接口设计

前言：

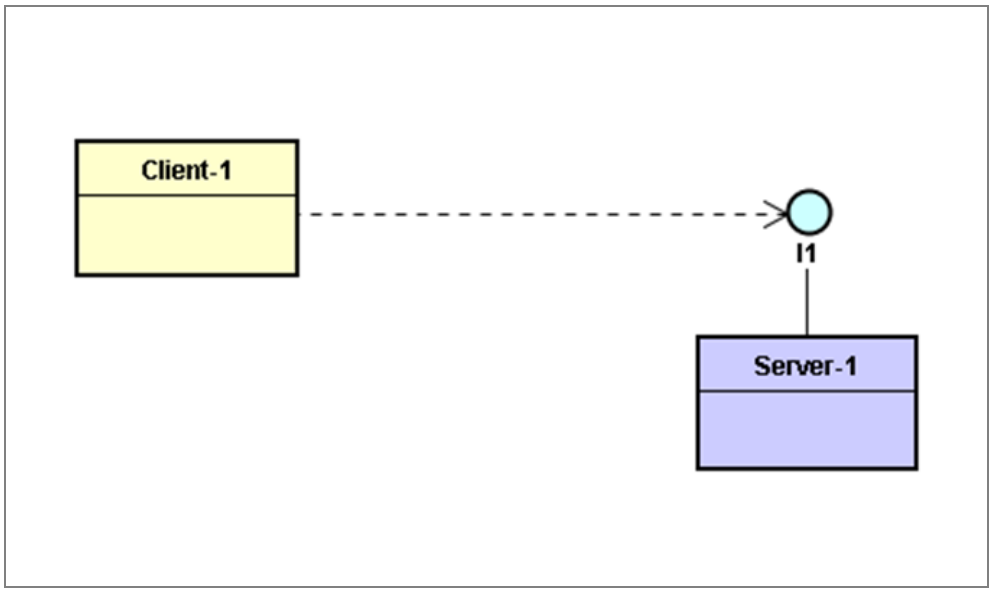
- 上一节针对 Server 端的<I>接口设计。
- 接下来，就来讨论针对 Cient 端的接口设计了。

4. 对 Client 端的通用性接口设计

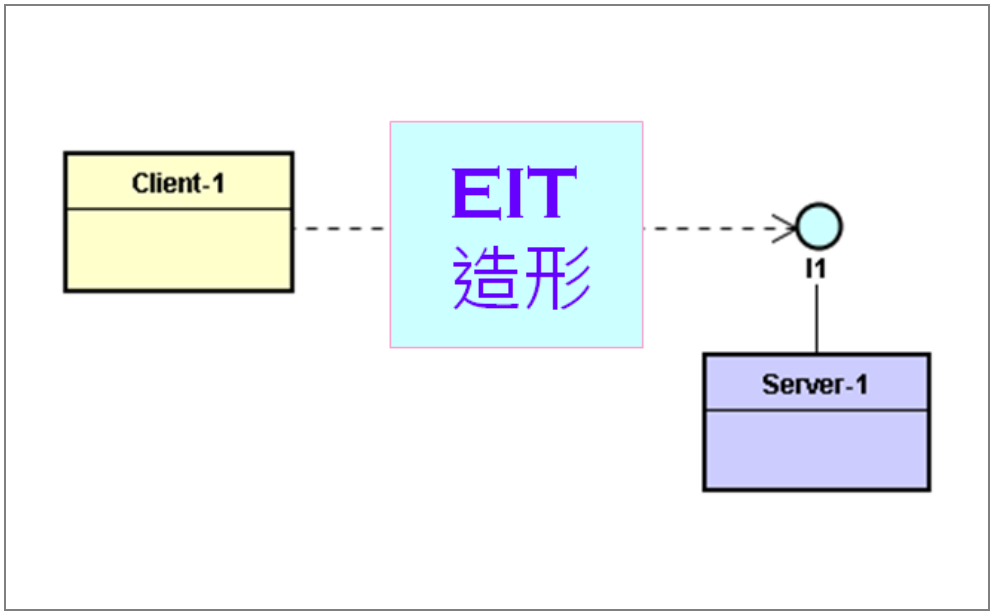
如果有 3 个团队各自开发自己的 Client-Server 模块，而且定义自己的接口。如下图：



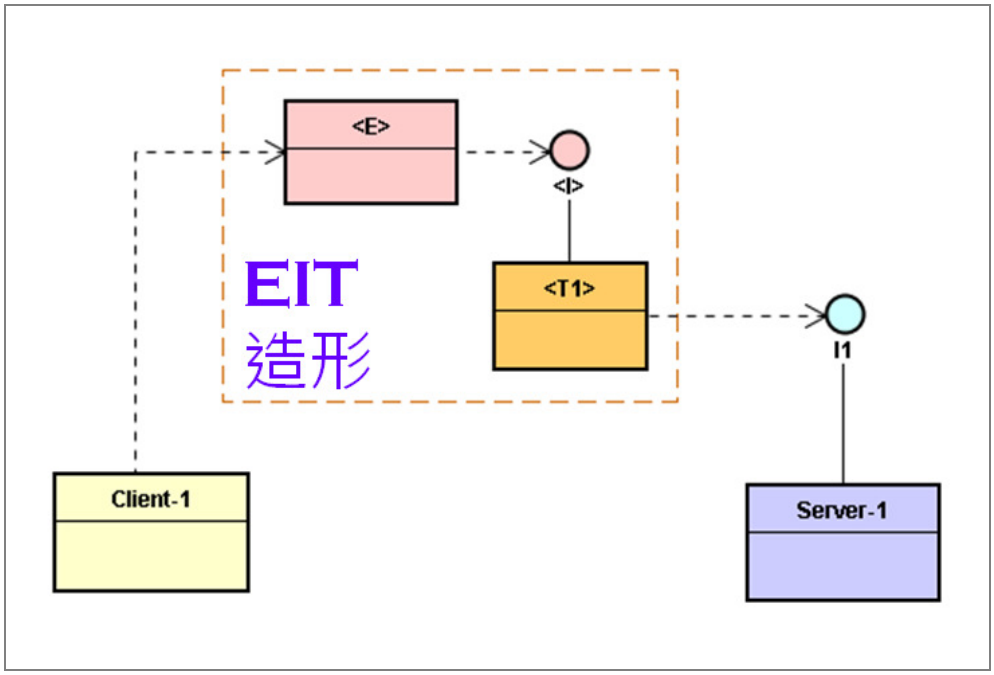
那么，如何将上图里的 I1、I2 和 I3 等接口统一起来，订定一个通用性的接口，提供给 3 个团队使用呢？答案是：使用 EIT 造形。首先，看看一个 Client-Server 结构：



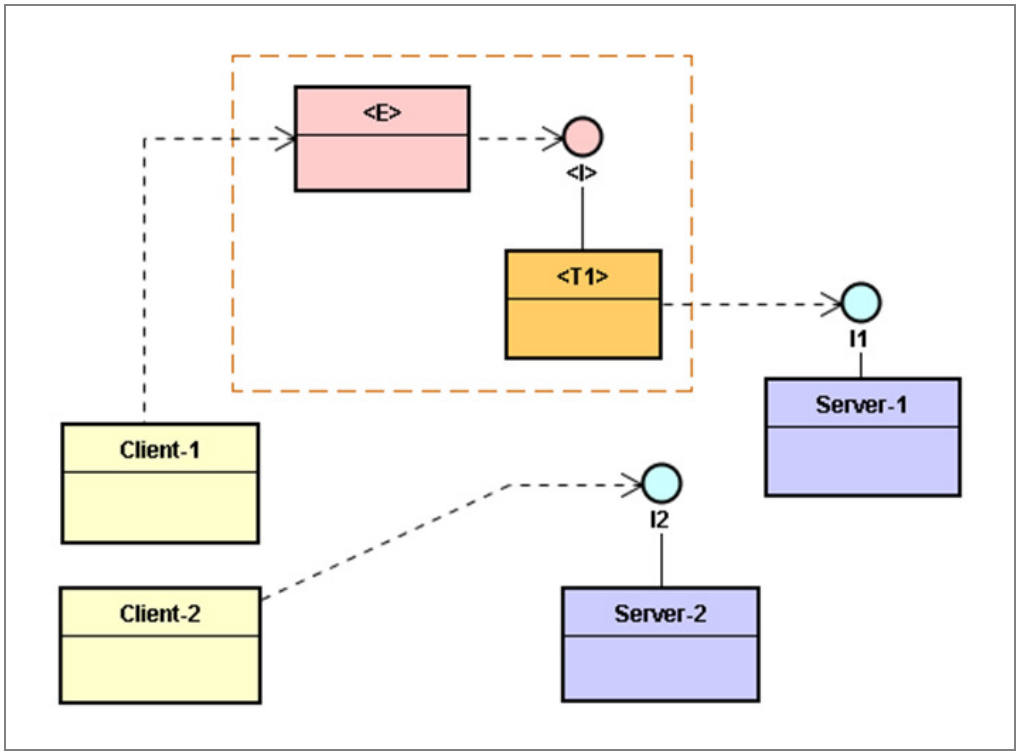
兹增添一个 EIT 造形，如下：



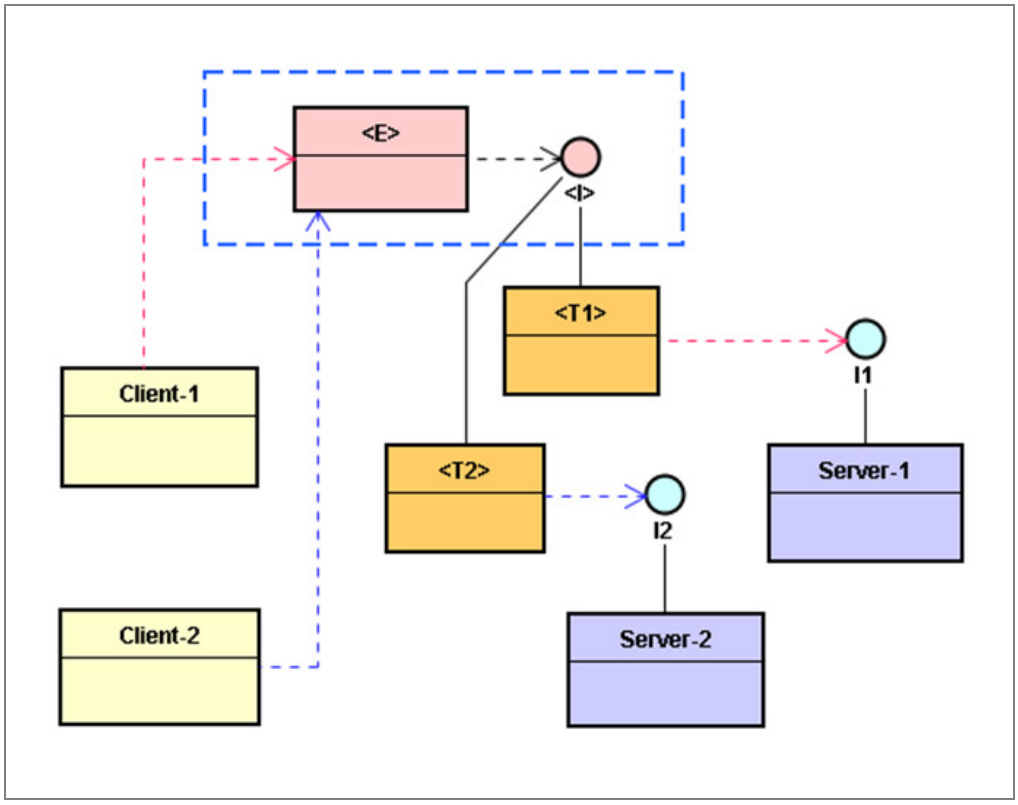
此 EIT 造形的内部结构，如下：



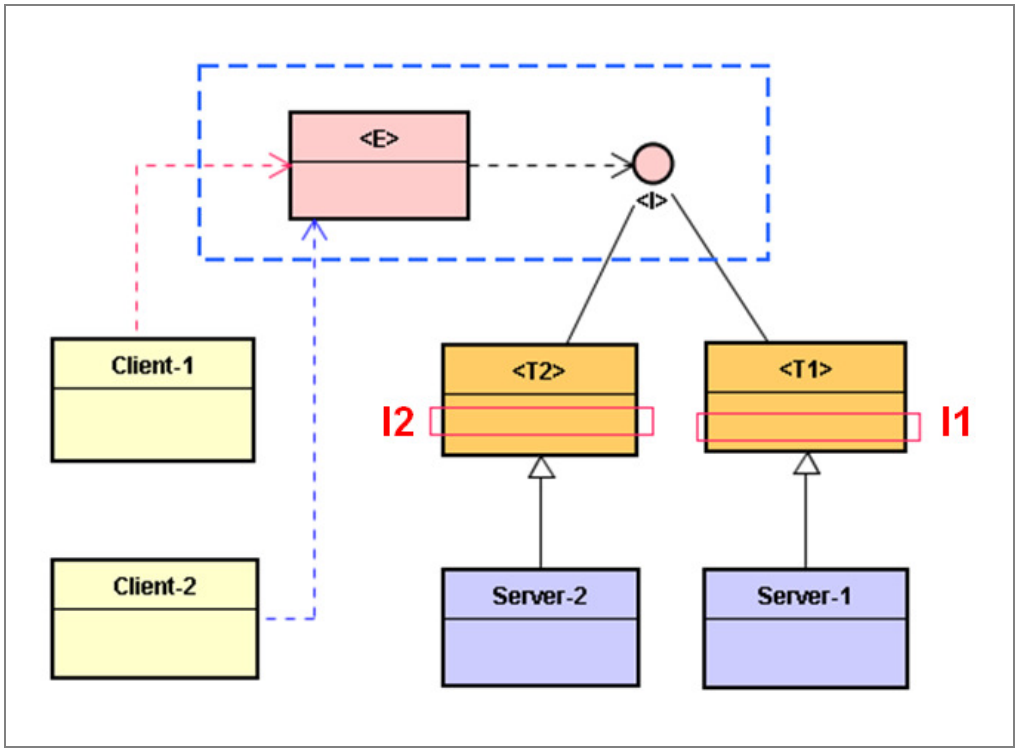
接着，继续考虑第 2 个团队所开发的 Client2-Server-2 架构，如下图：



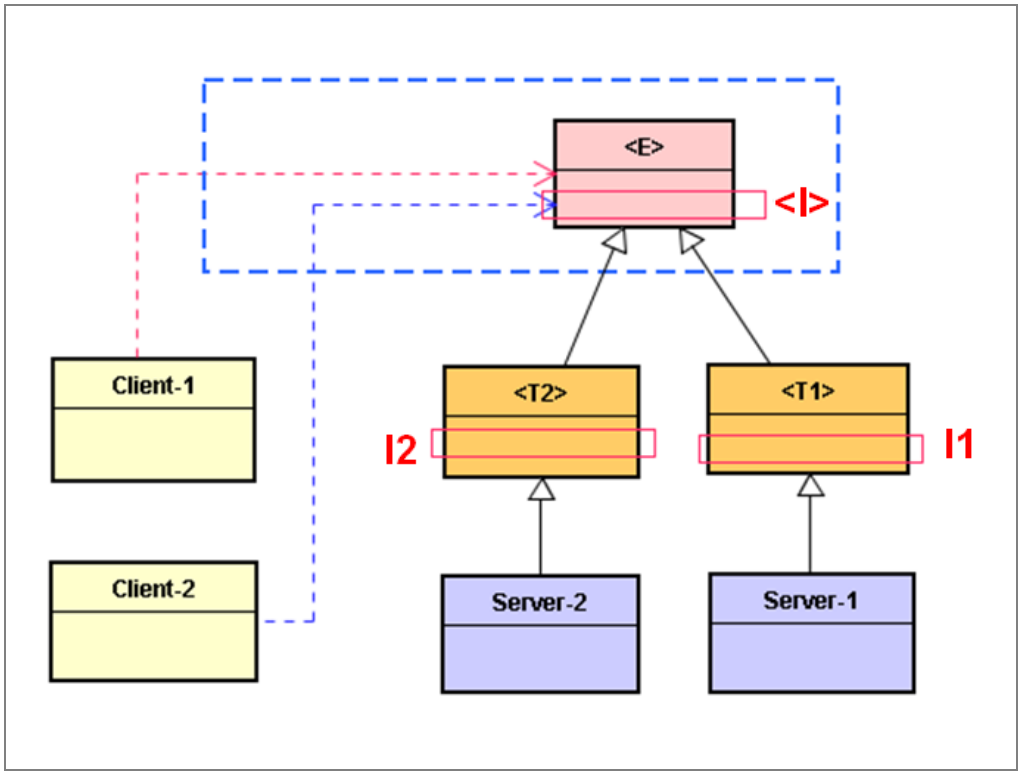
依样画葫芦，我们也能将 EIT 造形添加到 Client2-Server2 架构里，如下：



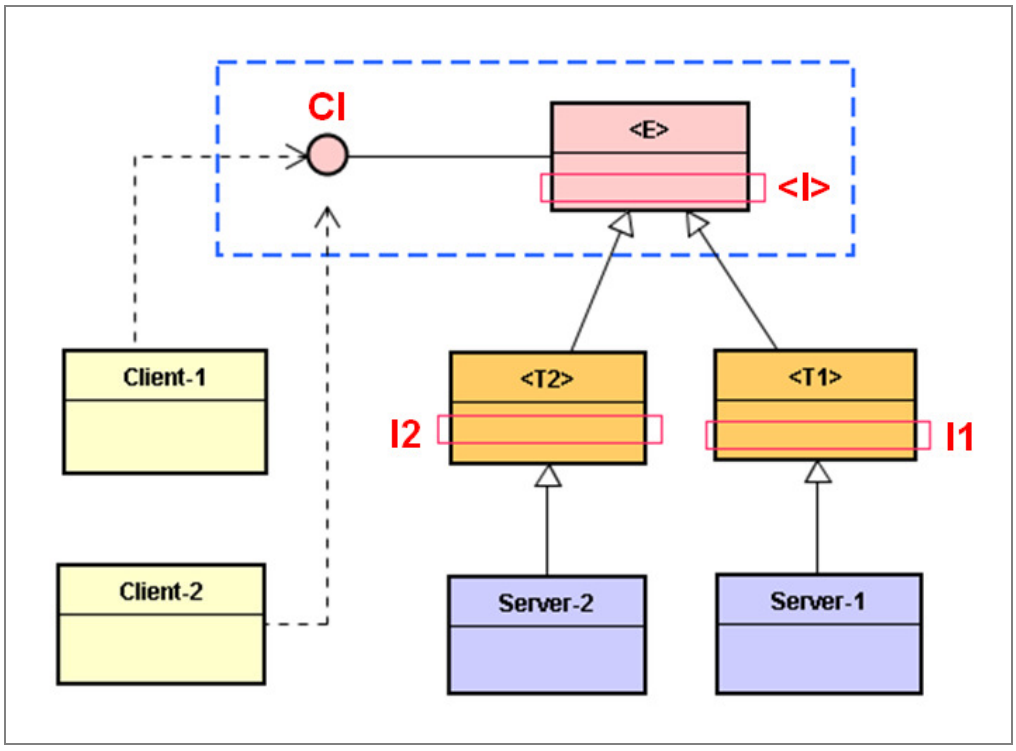
此时，两组 Client-Server 结构，共享同一组<E&I>了；此<I>就成为通用性接口了。 有时候，设计师会将接口 Ix 并入<Tx>类里面，如下图：



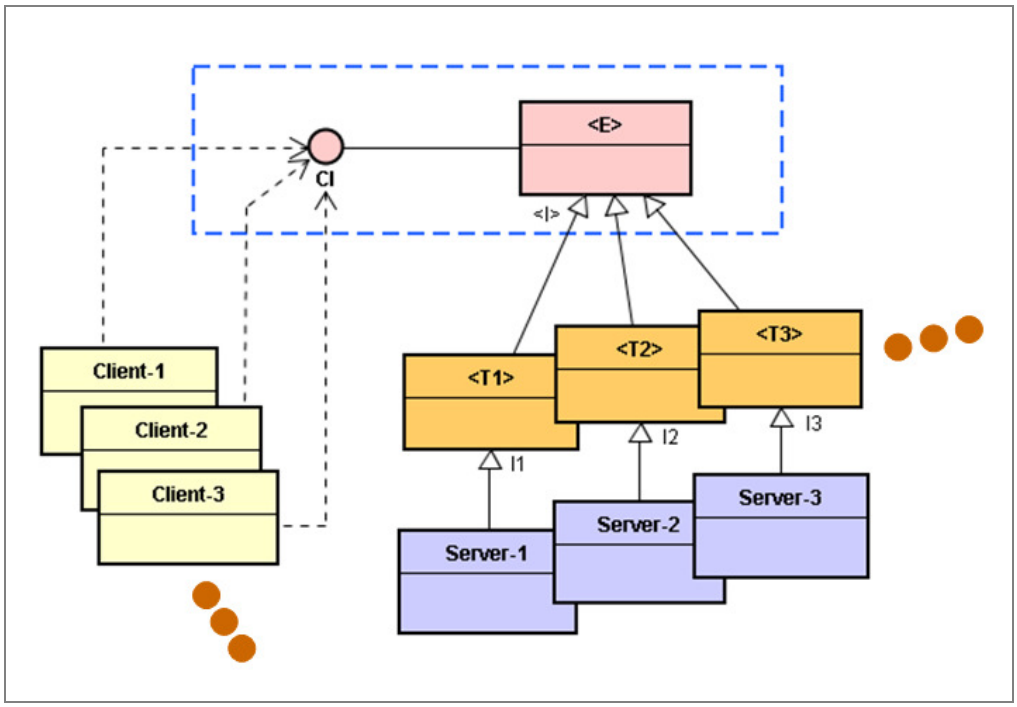
同样地，也可以将<I>与<E>合并，如下图：



接着，<E>也能提供一个 CI 接口，它是提供给 Client 来使用的公开(Public)接口；而<I>则是提供给<T>的通用性接口。如下图：



其中，CI 和<I>可以是一致的，也可已经由<E>的转换而有所不一样(如参数个数和型态等)。基于此项结构，Client 和 Server 个数都可以无限增加，而且都透过通用性的 CI、<E>和<I>来做为通信渠道。



于是，实现了统一的通用性接口的设计了。

在下一个单元里，将以实际案例来引导你更深刻体会通用性接口的妙用。

~ End ~