Chris Cargile	CSCI656	April 1, 2014	Ch.11 HW Questions

This homework assignment (plus reading) required 2 h. Answering questions from the assigned section required 1.5 h A recurring theme present in the textbook involves contrasting REST to HTTP architectures, so an additional .5 h was spent researching their differences based on the text's referenced literature and on the Skype architecture.

11.7.1) What are the consequences for a distributed application's architecture from realizing the network is *not* reliable?

At the cost of an application not being dynamically adaptable in the face of network failure, a distributed application may fail altogether. Several architectural strategies and styles used to fend off such risks are discussed in the chapter, such as those used by Napster, Skype, Gnutella, and BitTorrent. Although the Skype architectural design's decisions may have been a means of covertly circumventing the potential risk of application failure stemming from a single node in the network going down, the author's term its usage 'state-of-the-art,' and personally it has worked well for me (note the notes shared in this HW assignment's document header).

11.7.3) What are the 3 main stylistic elements from which REST was derived?

The REST architectural style is derived from 3 stylistic elements, leading to systems which are demonstrative of layered separation (in terms of Client/Server interactions, mobile code, or other architectural design elements), replication (of data/functionality), and limited commonality (often represented by using API's to represent transactions or system properties) properties.

11.8.1) For all six pair combinations of the following styles, describe how the styles can be used together at the same level of abstraction, or explain why they cannot: a) Layered/virtual machines b) Event-based c) Pipe-and-filter d) Rule-based

Using the graphics and descriptions of the 4 styles provided in Ch.4 as a guideline, the following is the response I am providing, given combinations:

<u>a/b)</u> The event-based style could or could not be paired with the layered/VM style in terms of being encapsulating depending on the flexibility and rules governing the styles' application to fill design goals, as shown using a 3-layered-architecture scenario. If the bottom 'layer' is a bus through which all events (on layer 2) communicate, the top layer (layer 3) consisting of events may not be able to communicate with those on layer 2 depending on whether layer 3 events are able to communicate using those on layer 2 as an intermediary. If layer 3 cannot use layer 2 as an intermediary (strict VM style), then there is no way for layer 3 to access the event bus thereby prohibiting it from exchanging message information vicariously.

<u>a/c)</u> Layered/VM style and P-and-F could be paired together, compatibly, if messages accumulate at a given layer to be piped for processing/filtration at the next layer. There would be a strict VM style

Chris Cargile	CSCI656	April 1, 2014	Ch.11 HW Questions

a/d) Layered/VM and Rule-based styles could be paired together, compatibly, if say the data/messages exchanged between layers were first/subsequently checked before proceeding to the next layer. If the layers were used in a way designed to yield a system which solved problems decomposable into simple predicate-based repetitive decompositions, this pairing would work suitably.

b/c) Event-based & P-and-F styles would not necessarily pair together suitably. If events can emit information and likewise have information piped into them, then the hub could be the 'connector' that facilitates such pipe-and-filter style processing of data.

b/d) Event- & Rule-based styles could pair together if deducing a problem from a known set of factual rules is achievable using communications between events across a hub. For instance, if the problem being considered was: "if CurrentItem is-a CarType, it must of MAKE 'german' or MAKE 'italian'" and the architectural event components included a: Database-interface, Main-thread/User-interface and rule-checker interface, the main-thread/UI 'event' could emit a query broadcast to the rule-checker interface via the hub to see if any rules existed for the CarType and upon learning one does exist, it could limit the breadth of its subsequent event broadcast query intended for receipt by the Database-Interface next. Thus, the rules-checker would act as a filtering mechanism to limit the breadth of database-queries, which are both enabled through the hub.

c/d) P-and-F & rule-based styles could be compatible if say problems are decomposable into repetitive type predicate-based decompositions. If, for instance, we knew only when A is-a B would we need to pipe the results from A into B, we could pair these two styles together to create a system whose overall style was derived from the unification of these two stylistic elements.

11.8.8) Describe the service architecture of a fast-food restaurant (such as In-N-Out Burger, etc) using the styles of Ch.4, in combination as necessary.

In the fast-food restaurant environment, messages taken from customers by the store-front and drive-through cashiers are enterable using pre-configurations keyable through the Orders kiosk (a computerized register using Rule-Based style elements) and are communicated through a PoS-to-Expediter linkage which is broadcast to multiple Terminals for display (using the BlackBoard stylistic element). Further, before generated orders are routed to the display terminals in the kitchen preparation area to be fashioned for delivery to the patrons, the appropriate food preparation clerk may be determined using pipes-and-filtering, such that Milkshake orders go only to the Dairy food-prep clerk and Burgers only to the Grill-station Food-prep clerk.