Health IT Standards Committee
A Public Advisory Body on Health Information Technology to the National Coordinator for Health IT

April 22, 2015

Karen DeSalvo, MD
National Coordinator for Health Information Technology
Department of Health and Human Services
200 Independence Avenue, SW
Washington, DC 20201

Dear Dr. DeSalvo,

The HIT Standards Committee (HITSC) gave the following broad charge to the Architecture, Services and Application Programming Interface (API) Workgroup:

Broad Charge for the Architecture, Services and API Workgroup:

The Architecture, Services and API Workgroup is charged with the defining of architectural patterns sufficient for an ecosystem of nationwide scale information sharing and modular applications serving patients, providers, provider-organizations, and researchers particularly as related to American Recovery and Reinvestment Act (ARRA) and the Affordable Care Act (ACA) which mandates a number of duties to the Office of the National Coordinator (ONC) relative to health information sharing. They make recommendations on standards, implementation guidance and certification criteria consistent with architectural patterns and make suggestions on how to achieve incremental progress towards proposed architectural patterns consistent with ONC roadmap and strategy. In close coordination with sister groups from HIT Policy Committee, they explore technology policy to promote the adoption and use of enabling technology consistent with the architectural patterns.

This letter provides recommendations to the National Coordinator, Department of Health and Human Services (HHS) based on the discussions that have taken place within the Architecture, Services and API Workgroup.

These recommendations were presented to the HITSC on Wednesday March 18, 2015 and formally approved by the HITSC on April 22, 2015.

Background:

Upon convening, the Architecture, Services and APIs Workgroup sought to create a framework for evaluating technology policy surrounding health information technology. The Workgroup believes that an informed framework on key architectural principles and patterns would be useful guidance for important

policy statements (such as the Interoperability Roadmap), standards and implementation guidance, and certification criteria.

By the term "architectural patterns", we did not seek to fully describe and fully constrain a proposed architecture plan, as might be the goal at an individual system or enterprise level. Instead, we describe *patterns* of interactions. We constrained those patterns to those that might work at the national scale of the nationwide health information exchange infrastructure described in the HITECH Act [ref HITECH and key definition] that links together a massive part of the US Health Care system. That infrastructure involves patients, providers of multiple types (e.g., physicians, hospitals, long term and post-acute care facilities, pharmacies, home health agencies, laboratories and radiology centers, etc.), associated Health IT systems, consumer and medical devices, etc.

To create a framework of architectural patterns for healthcare, we first looked to extant examples of ultra-large-scale system frameworks[1] that operate at a similar scale as US Healthcare. In particular, we examined the architectural principles and patterns underlying the Internet.

The Workgroup looked for architectural patterns in the literature documenting the key design decisions that informed the Internet. One of those critical design decisions was termed the Internet Hourglass, and was first described in the context of Internet transport [2]. Early transport stacks were layered and required close coordination between higher and lower level layers to achieve their aims. This layering and coordination limited adoption and scale of early networks. The engineering and standardization to create the Internet Protocol (IP) led to the description of the Internet Hourglass, depicted below.

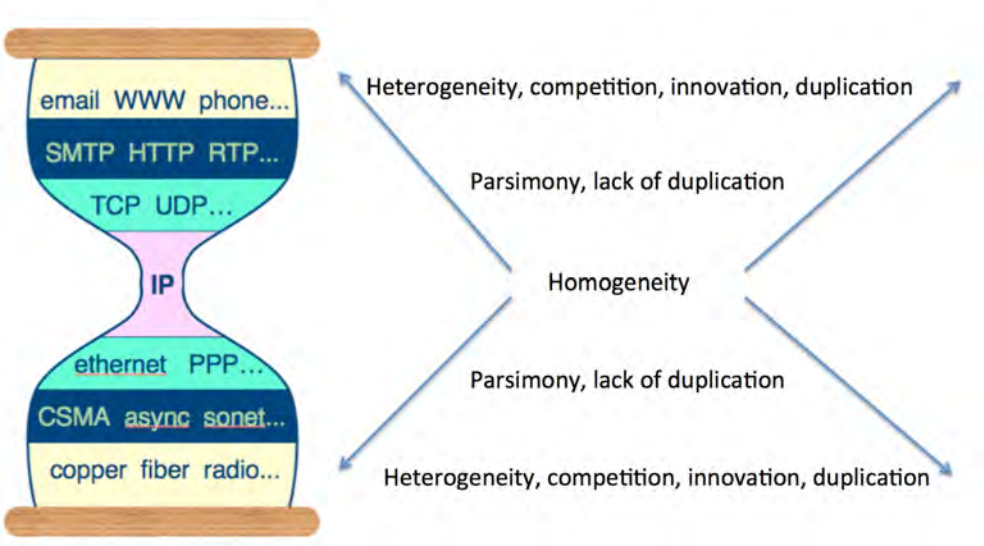

Figure 1: The Internet Hourglass

The Internet Hourglass is characterized by a homogeneous "narrow waist" of carefully designed core standards and a parsimonious set of ways of implementing and using that narrow waist, that allow variability and innovation both up and down the hourglass. The near seamless and uniform operational layer led to exponential growth in a competitive, multi-vendor marketplace.

As an example of how a smart choice for the narrow waist leads to a competitive and innovative marketplace, we can move down from the narrow waist and note that we can implement IP over multiple different network implementations, from coaxial cable and Ethernet over twisted pair, to the family of 802.11 wireless standards, to fully mobile standards such as LTE. Since all can implement IP (generally as TCP/IP) vendors could create networking models that solved for different business needs and competed on various attributes (e.g., scale, power consumption, TCO) without impeding large scale interoperability. Similarly, starting at the waist and moving up the hourglass, many different application protocols, including asynchronous protocols such as SMTP and XMPP, synchronous protocols such as HTTP, and real time streaming protocols such as RTSP can be implemented on top of the lower level TCP/IP protocol, which is itself layered over the narrow waist of the IP protocol.

The narrow waist emerges naturally from both good engineering design *and* strong ecosystem and network effects. The presence of a strong and scalable transport layer gives higher-level application protocol designers a powerful way to implement the application layer in terms of the lower layers. Open source implementations of the lower layers (e.g., the [Berkeley TCP/IP stack]) made it easy for operating system designers to build TCP/IP into their products. Ubiquity of TCP/IP gave hardware designers a large ecosystem into which to sell networking hardware.

The Workgroup observed these same network effects in other areas of the Internet and Web. For example, HTTP occupies for the web and modern web services the same narrow waist position as IP does for the broader Internet [3]. The emergence of the Web made HTTP the de facto transport protocol, and port 80 and 443 were commonly addressable through firewalls. Accordingly, designers of web services such as CORBA, XML-RPC, SOAP, etc. tunneled those services over HTTP transport. The popularization of the REST architectural style [4] made simple XML and later JSON representation over HTTP a viable services approach[1].

Because of the ubiquity of HTTP, the majority of modern programming languages and frameworks implement HTTP, enabling developers to use simple tools like CURL and browser plugins to inspect and debug content "on the wire." In contrast, higher level and more complexly coordinated web services (such

[1] In discussing high-scale systems, the Workgroup also noted that much of the scalability of the Internet comes from the end-to-end principle [11] that participants in a "conversation" do not have to pre-negotiate and maintain independent or central knowledge of sender and receiver "state." The REST style as implemented in HTTP fulfills the end-to-end principle by transmitting representations of resources in transactions, which greatly eases the burden of building ultra-large-scale systems.

as SOAP, or CORBA) require significant development effort for libraries and tooling. These libraries and tools are available “out of the box” on some languages and frameworks (e.g., Enterprise Java and .NET) but are not available on many others (e.g., Python and Ruby, newer programming languages like Go and Rust, and importantly, on mobile platforms such as iOS, Android and Windows Mobile). This, in turn, cements “good enough” simple JSON and XML representations flowing over HTTP as the near-universal approach. Services and cloud providers make their services available natively as simple XML or JSON representations over HTTP; framework developers support this style across all platforms, driving a similar ecosystem and network effect favoring HTTP as the universal Web “transport”.

These observations suggest to the Workgroup that the Hourglass Pattern more generally involves the following:

- The narrow waist of homogeneity and parsimony must occupy a position where it may be implemented across a wide range of systems and be used to implement a wide range of higher-level applications.
- Accordingly, the narrow waist must be “good enough” to be used in a variety of situations with reasonable performance and quality.
- The narrow waist drives powerful positive ecosystem and network effects, such that the presence of sufficient numbers of implementations (including open source libraries) drives applications, and large numbers of applications drive additional and better implementations.

The presence of these network effects and “good enough” homogeneous and parsimonious standards ensures that even if there is a theoretically “better” way of implementing a single interoperability use case, the greatest ecosystem benefit accrues to implementing interoperability in terms of re-use of the capabilities provided by the narrow waist.

The Health IT Hourglass

The Workgroup then considered application of the Hourglass pattern to health IT. The Workgroup looked at examples of successful and attempted standards-based ecosystems in health IT, including HL7 V2 messaging, NCPDP messaging, the HL7 V3 ecosystem, the ebXML family of standards used in public health, IHE document exchange profiles, HL7 FHIR, and the SMART on FHIR pluggable application approach.

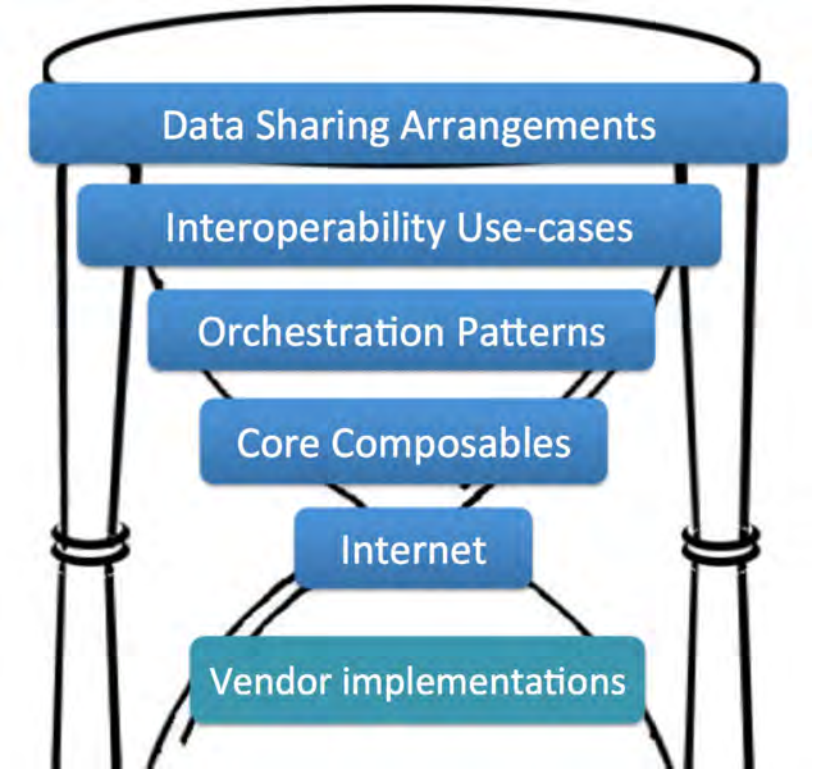

Figure 2: Our Proposed Health IT Hourglass

Core Composables

We observed that successful Health IT standards create modular reusable data elements that address common needs for data exchange and can be combined (or "composed") together to address particular use cases. For example, the very successful and long-lived HL7 V2 standard defines a set of common health care data types (e.g., identifiers, dates, etc.), and segments composed of fields and data types that address common exchange needs (e.g., patient and provider identifying information, clinical observations, etc.). These fields and segments are further composed together to create message types, which are then constrained to address use cases, such as Admit/Discharge/Transfer workflows or laboratory orders and results workflows.

Composability is based on the principle that each composable element can readily be combined with other elements, such that complex functions can be deployed by creating simple combinations of the core composables. Composability enables modularity of design and reuse of key assets, resulting in myriad potential implementations and use-cases without requiring the "start from scratch" approach often seen with "bespoke" approaches to interface design.

We have named this the **Core Composables** pattern. We have identified four initial classes of composable elements: data access, transport, user experience, and security. We recognize that more classes may emerge with additional analysis.

Data Access Composables

The data access composables define the building blocks for reusable data access , and consist of:

1. A set of core data elements and simple data structures that represent the most common data that is transferred in health care ("data resources")
2. A simple way to create defined extensions and additions to those data resources to address specific or less common data needs
3. A way to compose data resources together to create composite or aggregate data needed to represent more complex exchange scenarios (for example, a laboratory report, a listing of current medications, or a summary of current health status)
4. A mechanism to constrain data resources and composites to establish conformance to a defined exchange use

Transport Composables

1. Services that enable modern, mobile-friendly resource access (simple XML or JSON resources with transport over HTTP [4], that expose CRUD (Create, Read, Update, Delete) operations on the composable data resources.

User Experience Composables

1. A way to define and expose user experience / user interaction as a core building block. If necessary for a given use-case, this composable can be used to display data or graphics, capture user input, and to create programmatic effects such as interactive form navigation, etc.

Security Composables

1. Standard protocols that implement the most common authentication use cases, such as single sign on (SSO)
2. Standard protocols that implement the most common authorization requirements, such as peer-to-peer authorization, third-party proxy authorization, and user-mediated access capabilities

Candidate Standards for Core Composables

We have compiled a list of proposed candidates for the four classes of Core Composables.

Data Access

The Workgroup found that **HL7 FHIR** [5] is a promising emerging standard that explicitly addresses the requirements of the Core Composable data access pattern. FHIR, when combined with adequately constrained FHIR Profiles, can create simple but powerful composable data access services. FHIR is now undergoing extensive development and testing.

Transport

The Workgroup found that modern use of **HTTP** addresses both scalability and composability for CRUD data access services and does so in ways that are more open to mobile and web ecosystems than does enterprise-oriented standards like SOAP or CORBA.

User Experience

The Workgroup found that **HTML5** can capture and convey a comprehensive set of user-experiences and interactions. HTML5 is supported by a wide variety of devices and platforms, and can serve as a baseline capability for near-universal deployment of visual interactions, if such capability is called for by a particular use-case. We consider CSS and Javascript to be part of the HTML5 composable.

Security

The Workgroup recommends **TLS** and X.**509** certificates for basic secure channel communication of HTTP transactions. We recommend **OIDC** for single-sign-on style authorization, and **OAuth 2.0** for third-party proxy authorization. We recognize that this area will need more analysis to cover additional authentication and authorization models.

Orchestration Patterns

The workgroup also discussed the need to address the common workflows (sequences of interactions) by which data are exchanged in healthcare. Our goal was to decompose these common workflows into simple combinations of Core Composables. We observed that a few common interaction patterns can account for many and varied use-cases. We elected to call attention to this important principle by naming it **Orchestration Patterns** ("Orchestrations" for short.) The recognition that a small number of well-defined orchestration patterns could leverage the Core Composables to address a wide variety of use-cases is a key finding of the Workgroup. Orchestrations can capture reusable interaction patterns in much the same way that the core components capture reuse at a lower level of the Health IT Hourglass.

There are many common system interactions that we believe can be formalized into specific Orchestration Patterns for use in Health IT. For example, asynchronous message queuing, by which one system notifies another of a critical event so that the second system can account for the event, is a ubiquitous orchestration pattern in the current system for health information exchange and is heavily used with HL7 V2 messages. A related exchange pattern is the publish-subscribe messaging approach, by which one system broadcasts an event or message, which is then asynchronously delivered to other systems that have registered their interest in that message. As an example of this pattern, ADT-based admit and discharge notification services are a popular health information exchange service, by which members of a patient's care team are notified whenever the patient is admitted to or discharged from an acute care or emergency care setting.

A fully developed Orchestration Pattern is partly or wholly independent of the Core Composables that are used to implement the pattern. For example, a message queuing pattern may define common data content to identify message acknowledgement or message topics. The Orchestration Pattern should define the basic transport and security needs required for developers to implement the orchestration. However, the actual message content passed from one system to another is wholly independent of the messaging Orchestration Pattern.

As an example of this, the Workgroup considered an emerging and important new Orchestration Pattern: the **Pluggable App** pattern. In this pattern, the Core Composables include HTTP, TLS, OAuth 2.0, FHIR, and HTML5. These core components are orchestrated by the Pluggable App specification such that a wide variety of "apps" can be created that can then be "plugged in" to any compliant Health IT system. In this pattern, an application (for example, an HTML5-based web application running inside a web browser embedded in the EHR workflow or on a mobile device) can use OAuth 2.0 to gain appropriate authorization on behalf of a patient or provider to access health data using appropriately profiled FHIR Core Composables.

Combining Core Composables and Orchestration Patterns to address heterogeneous use cases

Having implemented Core Composables and at least one Orchestration Pattern, it should be possible for developers to readily address new uses cases by reusing the underlying Orchestration Pattern. For example, the following use cases are potential use-cases for the Pluggable App pattern:

- Query a Prescription Drug Monitoring Program (PDMP) to provide decision support on the risk of drug seeking behavior, given a patient and prescription order
- Evaluate the risk that a patient has been infected with a disease that may be an emergent epidemic (e.g., Ebola decision support)
- Provide specialized disease-specific decision support, documentation or novel visualization
- Collect differential information that may be required for a clinical trial
- Display and reconcile EHR sourced information with that from a health information exchange or population management registry

This ability to reuse a simple Orchestration Pattern for a variety of heterogeneous use-cases has significant implications for future approaches to expanding interoperability for Health IT. By implementing the core composables and then supporting common Orchestrations, Health IT vendors should see an increase in the number of achievable interoperability goals, with less work than crafting purpose-specific interfaces. The Workgroup also believes that re-use of common Orchestration Patterns can become an efficient way to address the need for more innovation in Health IT, as was suggested by the JASON report [6] and echoed by the Jason Task Force [7]. For example, the Pluggable App orchestration pattern could enable Health IT systems to become “platforms” against which a variety of independently-authored apps can be deployed, thus reducing the vendor customization needed for each app. Having implemented the Pluggable App pattern, a Health IT systems can use the pattern to fulfill a number of use cases by leveraging the capabilities of the underlying Core Composables.

As an example implementation of the Pluggable App pattern, the Workgroup noted that the SMART on FHIR work [8] performed under ONC SHARP grant funding has published a open specification for implementing a Pluggable App orchestration. A simplified exposition of the actual Orchestration proposed by SMART on FHIR is shown below to serve as an example of a non-trivial Orchestration. Examples of Pluggable Apps are provided at [9].

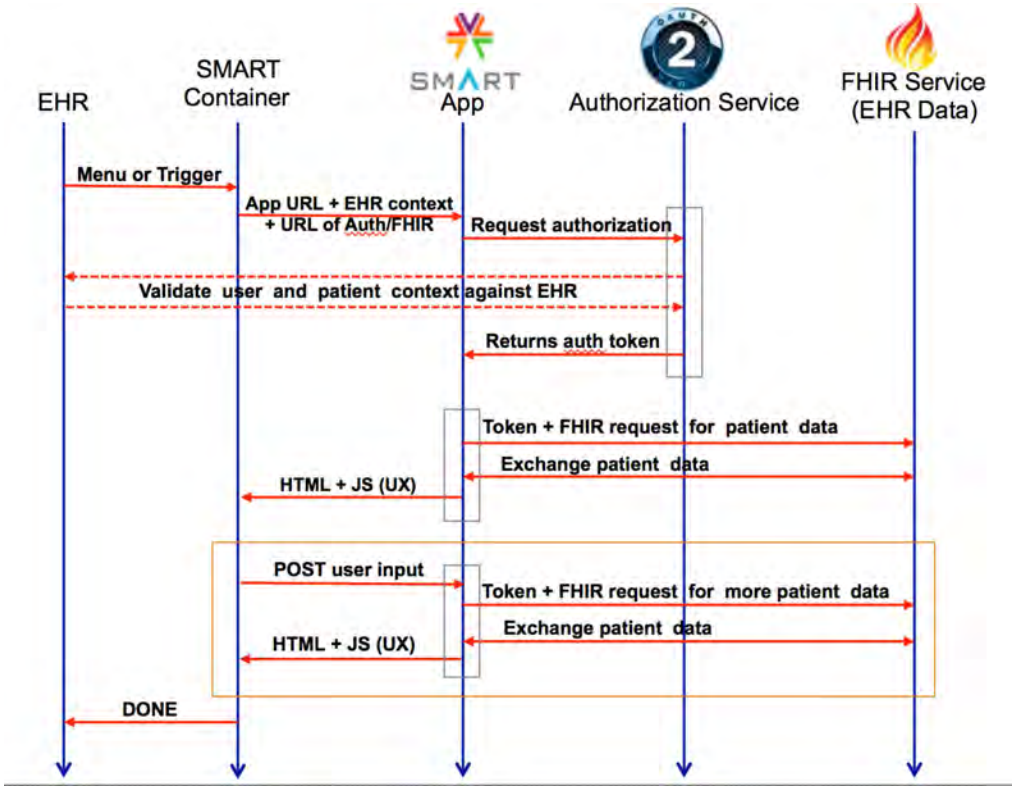

Figure 3: Example Pluggable App Orchestration

The API Workgroup believes that there are a number of other key Orchestration Patterns that should be defined by the Health IT community to capture other reusable integrations of Core Composables. For example, "**Clinical Decision Support as a Service**" (CDSaaS) is a potential extension of the Pluggable App orchestration. In CDSaaS, triggers based on actions performed in the EHR could invoke a remote CDS service by first authorizing a composable peer-to-peer conversation. In the background, the remote CDS service queries the EHR for any necessary additional data by using composable FHIR resources, and, if necessary, asks the EHR to "pop up" a Pluggable App to capture any additional data from the provider, and/or to provide decision support advice to the clinician. The Workgroup believes that there are numerous CDSaaS vendors who would be able to leverage this type of Orchestration Pattern, thus simplifying and facilitating

EHR adoption. We also speculated that this approach would be more flexible and open-ended than the current Health eDecisions proposal [10].

The Workgroup compiled this illustration of potential uses of this "Health IT Hourglass" framework. The listed items should be taken as examples. Some of these examples will require additional investigation before become widely accepted patterns and use-cases.

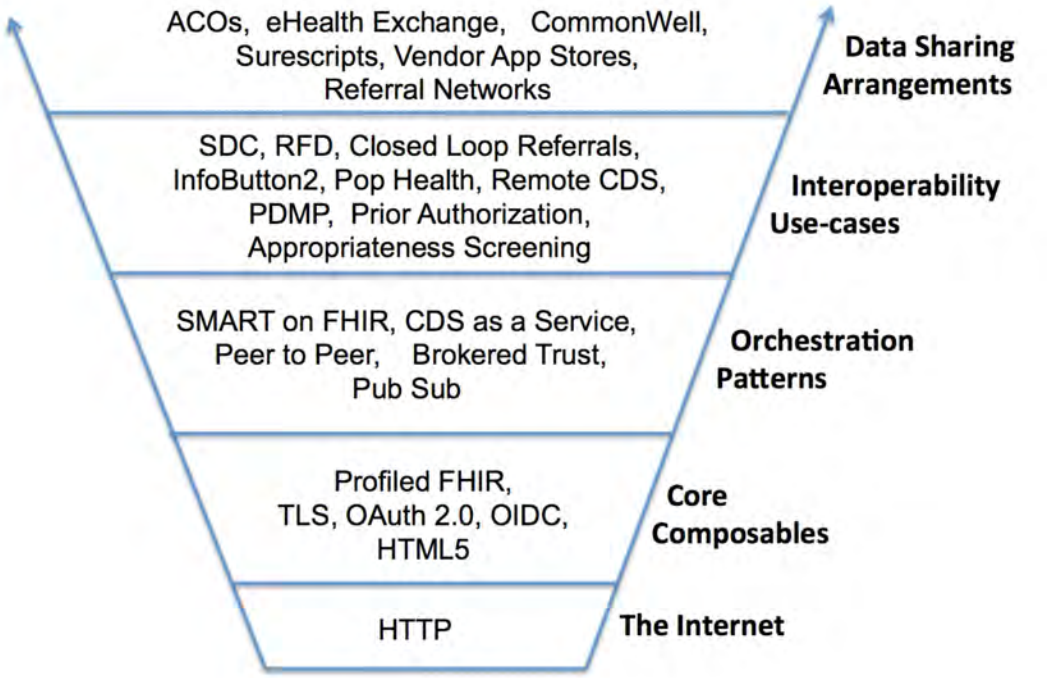

Figure 4: Example Health IT Hourglass Instantiation

Recommendations

Based on the Health IT Hourglass framework, the Workgroup formulated a set of recommendations for ONC.

Recommendation: Define a roadmap towards the Health IT Hourglass

1. To create greatest modularity, move towards parsimony of
    - For Transport and Security Composables: HTTP, TLS, X.509, OAuth 2.0, and Open ID Connect

- For Data Composables: Profiled FHIR, implementing the Common Core Data Set, expanding to include important Profiles for precision medicine, patient assessments, and key specialties over time.
- For interaction Composables: HTML5, including CSS and Javascript, for cases where a universal encapsulation of user experience is needed.
- Orchestration Patterns: Starting with Pluggable Apps, extending with other identified common Orchestration Patterns such as CDS as a Service.

2. Adopt a deliberate policy of "rebalancing" the standards portfolio towards the Health IT Hourglass model
3. Allow sufficient time to develop, adopt, and use Core Composables and Orchestration Patterns to allow for demonstrations of success during the rebalancing period
4. As recommended in the joint Health IT Policy and Health IT Standards Committee recommendations from the JASON Task Force [7], provide flexibility for detailed policy governance of specific use-cases to be performed by Data Sharing Arrangements

Recommendation: Identify critical priorities for 2015–2017

1. Create a glide-path to Core Composables and Orchestration Patterns by
   - Supporting SDO and public-private (e.g., Argonaut, S&I DAF) work to define core composable API services and profiles
   - Supporting SDO and public-private work to define orchestrations and related security components for complex orchestrations such as Pluggable Apps
   - Supporting future work to define other key high value orchestrations and security components (e.g., Peer to Peer record access, CDS as a Service, Pub Sub)
   - Supporting priority use-cases work (e.g., CDS, PDMP, SDC, etc.) to be implemented in terms of Core + Orchestration Patterns
2. Reduce "friction" and distraction to adopters and implementers by
   - Minimizing certification requirements overall to allow ample time to pilot, adopt, and refine Core and Orchestrations
   - Ensuring that government incentives can be met using the newer approaches, even if not formally adopted into Certified HIT.
   - Continuing to support production adopted standards not based on Core Composables (e.g. C-CDA, XCA/XCPD, etc.) while minimizing changes and new uses of non-composable standards
   - Avoiding endorsing new standards that are not based on Core and seek alternatives that are based on Core (e.g., HPD+, CSD, HIEM)

Recommendation: Identify roadmap priorities for 2018–2020

1. Refine and extend core composable services, profiles, and orchestration patterns
2. Expand the number of piloted use-cases based on the core composable model
3. Address needs for national-scale services such as MPI, RLS, Directories, etc.
4. As Data Sharing Networks emerge, address needs for network-bridging services
5. Consider mature APIs, orchestrations, and use-cases as candidates for addition to Certified HIT
6. Begin transition from non-Core/Orchestration standards and APIs

Recommendation Identify roadmap priorities for 2021–2024

1. Continue work from the 2018–2020 period and:
    - Address complex data profiles that require more robust data models (e.g., CIMI) as may be needed for the Learning Healthcare System
    - Contemplate transition to new Core/Orchestrations based on the- current technology directions

References:

[1] L. Northrop, R. P. Gabriel, D. C. Schmidt, and K. Sullivan, *Ultra-large-scale systems*. 2006.

[2] B. E. Carpenter, "Architectural Principles of the Internet," 1996. [Online]. Available: https://tools.ietf.org/html/rfc1958. [Accessed: 17-Apr-2015].

[3] L. Popa, A. Ghodsi, and U. C. Berkeley, "HTTP as the Narrow Waist of the Future Internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, vol. Hotnets, pp. 1–6.

[4] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000.

[5] HL7, "FHIR," 2014. [Online]. Available: http://www.hl7.org/implement/standards/fhir/. [Accessed: 18-Dec-2014].

[6] JASON, "A Robust Health Data Infrastructure," 2014. [Online]. Available: http://healthit.gov/sites/default/files/ptp13-700hhs_white.pdf. [Accessed: 21-Apr-2015].

[7] J. T. Force, "JASON Task Force Final Report," 2014. [Online]. Available: http://www.healthit.gov/facas/sites/faca/files/Joint_HIT_JTF Final Report v2_2014-10-15.pdf. [Accessed: 21-Apr-2015].

[8] "Something New and Powerful: SMART on FHIR® | SMART Platforms," 2014. [Online]. Available: http://smartplatforms.org/smart-on-fhir/. [Accessed: 21-Jan-2015].

[9] “SMART App Gallery,” 2015. [Online]. Available: https://gallery.smarthealthit.org/. [Accessed: 17-Apr-2015].

[10] ONC S&I, “Health eDecisions Homepage,” 2013. [Online]. Available: http://wiki.siframework.org/Health+eDecisions+Homepage. [Accessed: 17-Mar-2015].

[11] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Transactions on Computer Systems*, vol. 2, no. 4. pp. 277–288, 1984.