



# **Committee on the Review of the Adoption and Implementation of Health IT Standards by the DHHS Office of the National Coordinator for Health Information Technology**

**Charles N. Mead, MD, MSc**  
*Chief Technology Officer*  
*National Cancer Institute*

**Senior Associate**  
**Booz Allen Hamilton**

# “Interoperability”

- Achieving interoperability is hard – achieving *computable semantic interoperability (CSI)* is very hard
  - CSI: what applications in which domains in which orgs?
    - Separation of Use Cases (or aspects of Use Cases) that require *computable* semantic interoperability from *human* semantic interoperability
  - “there is an important role for text...”
    - HITSP identification of CDA
      - Facilitates “incremental CSI” by enabling data/information exchange between systems of different “information maturity”
- Clear, comprehensive identification of all the project’s stakeholders is a *critical success factor*
  - Who mandates, pays for, uses (primary and secondary), builds, and regulates the (proposed) solution?
- There are no shortcuts in this process – Find the time (and money) to do it right or make the time to do it over.

# Software Engineering Facts

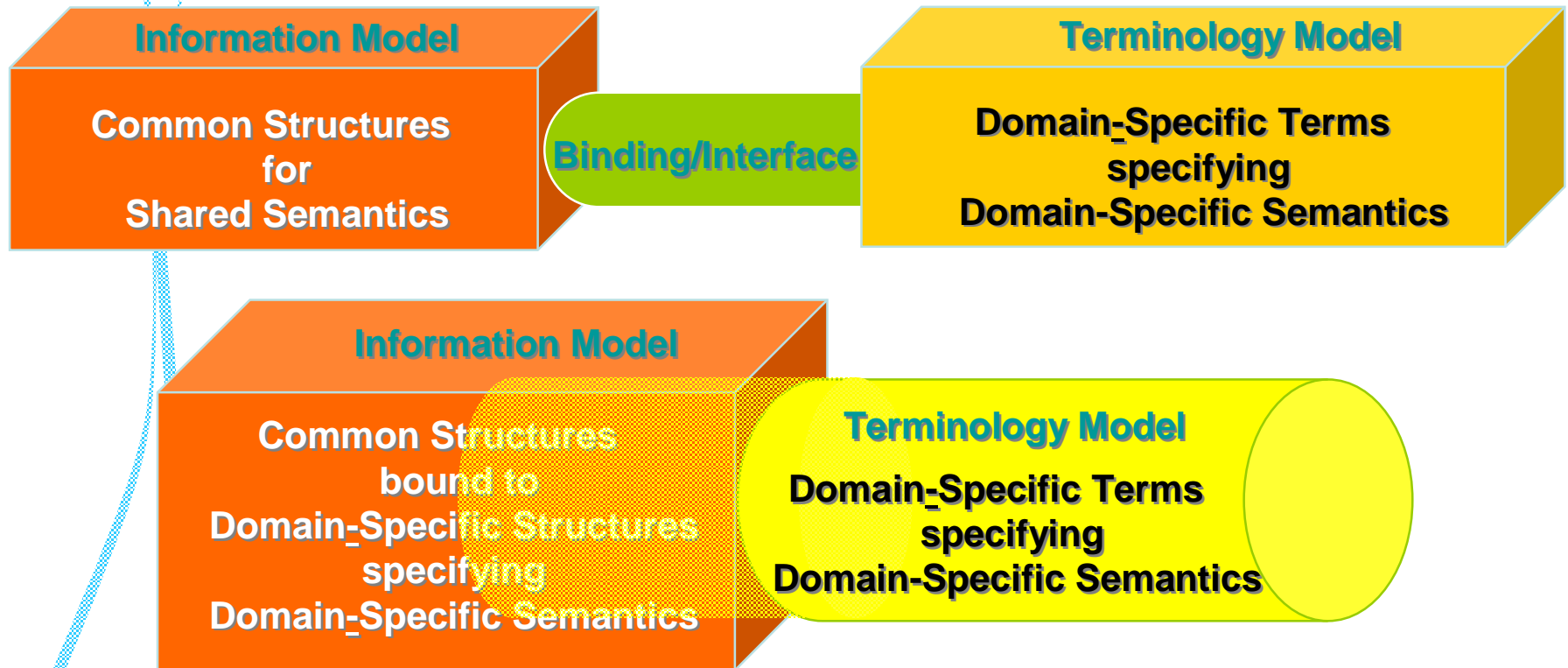
- Large projects fail often
- Complex projects fail often
- This is a large and complex project
- Best Practices
  - Multidisciplinary teams
  - Ongoing risk identification and management
  - Iterative/Incremental process
  - Architecture-centric process
- Iterations are critical for a complex project's success – failure is certain under Waterfall (“gather all the requirements first”)
- *Abstraction* and *Layering* are essential approaches – resist the temptation to dive too deeply too soon
- Recognize that differences in process do not necessarily mean differences in static semantics

## “Minimum Architectural Impact”

- “Architecture” – structure + function, an implementation/solution of a specified problem that represents a set of compromises
- Use Cases define functional requirements and (particularly in the case of the HITSP UCs) static data that must be passed between systems
  - Architectures of existing systems may *have* to change to support UC-defined *semantics*
    - e.g. change of existing database schemas →
    - application rewrites →
    - associated data migration strategies
- It is possible (and desirable) to be *implementation neutral*, but it may not (probably won't) be possible to *architecturally neutral*
- In addition, a number of “quality requirements” (e.g. security, performance, etc.) are often ‘hidden’ in UCs (and are often UC-specific)

# Information vs Terminology Models

## Intersecting and interleaving semantic structures



*Example:* Appropriately constructed semantic web structures should be able to distinguish between “Grade IV allergic rxn to Penicillin” represented in several ways using various combinations of RIM and SNOMED-CT codes.

# A caBIG™ Example:

## *Achieving CSI within the CTMS WS*

- Interoperability requirements driven by stakeholders (storyboards à Activity Diagrams à Use Cases à Domain Model)
- Shared model of domain semantics (BRIDG “information model”)
- Existence of workspace-wide multi-disciplinary team (CTMSi team) for interacting with/overseeing all CTMS development
- Required harmonization of all CTMS applications with BRIDG model (“architectural neutrality”)
- Binding of BRIDG to V3 data types
- Existence of meta-data repository and associated terminology repository to manage binding of value sets (“standards”) to information model concepts (caDSR)
- Registration of information model in caDSR
- Wire format based on BRIDG and caDSR semantics
- Registration of application APIs in meta-data repository
- Certification process for each application (caBIG™ Compatibility)

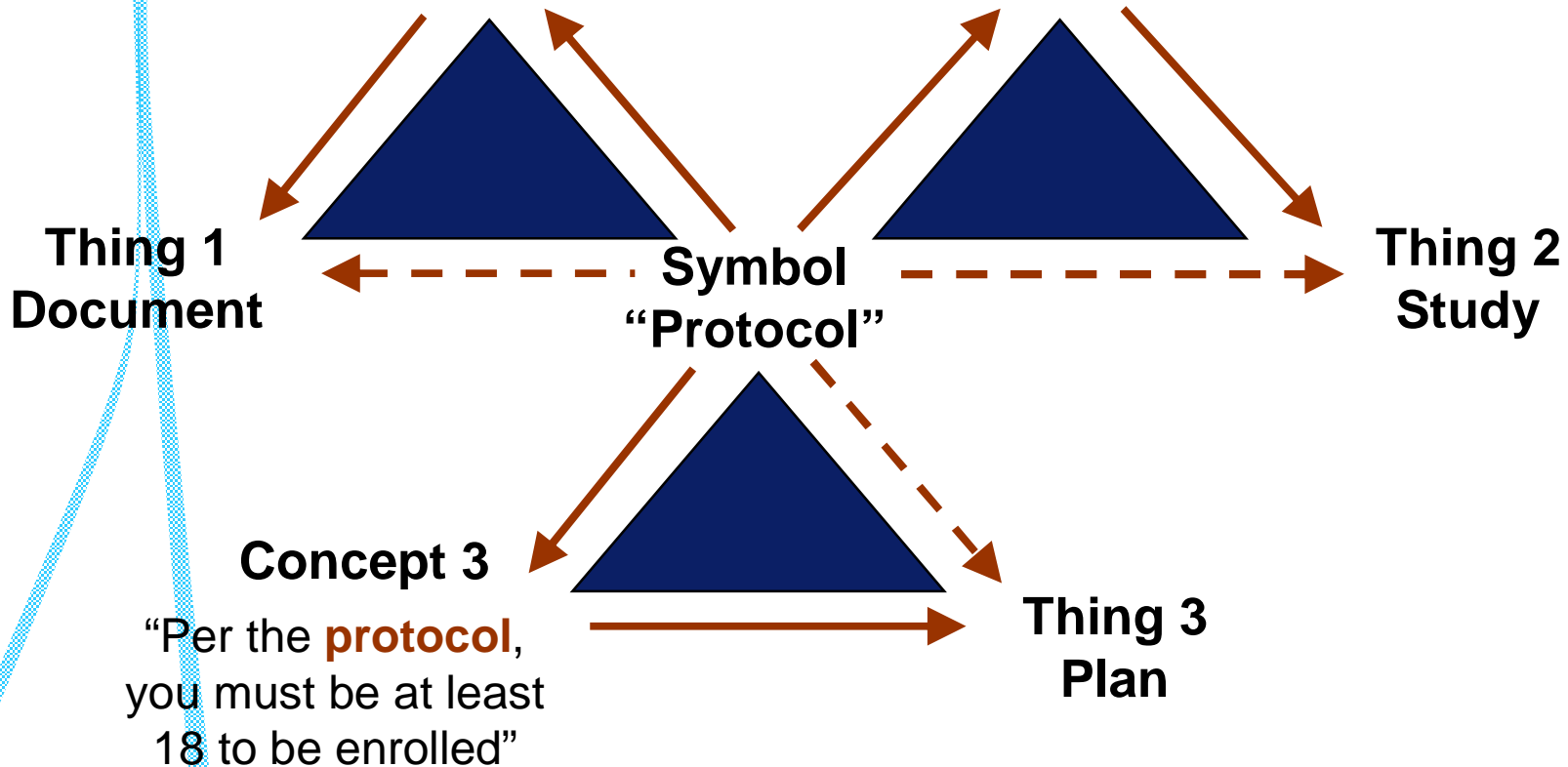
# “Protocol” – a ‘common’ term...

## Concept 1

“We need to sign off on the **protocol** by Friday”

## Concept 2

“**Protocol** XYZ has enrolled 73 patients”



# Summary

- The goal and work of AHIC, ONC, and HITSP in defining and deploying standards is certainly a positive step forward in achieving the goal of developing an nationwide framework to support interoperability.
- Ongoing success will require continuous recognition of the overarching complexity of the problem being solved, particularly in terms of utilizing the appropriate tools, processes, and frameworks supported by the appropriate amount of funding and expectation setting.
  - It is not a question of speed *per se*, although speed it often a seductive metric...



eHealth Interoperability  
in Europe

# Levels of Interoperability

<b>Health policy:</b> <u><i>cooperation</i></u>	<ul style="list-style-type: none"><li>• Vision &amp; strategies</li><li>• Processes &amp; measures, incentives</li><li>• Socio-economic (sustainable), legal framework</li><li>• Accreditation and certification</li></ul>
<b>Health service providers (Organisational level):</b> <u><i>collaboration</i></u>	<ul style="list-style-type: none"><li>• Organisational structures and culture</li><li>• Intra &amp; inter-jurisdictional service processes</li><li>• Change management, behavioural change</li><li>• Systems thinking, business process re-engineering</li></ul>
<b>Semantic</b> <u><i>interoperation</i></u>	<ul style="list-style-type: none"><li>• Terminologies, classifications</li><li>• Translation</li></ul>
<b>Technical / functional</b> <u><i>interoperation</i></u>	<ul style="list-style-type: none"><li>• Technical standards</li><li>• Hardware and software connectivity</li><li>• User interfaces</li></ul>

## ***LINUX and WebDAV:***

### **Two Large and Complex Projects that have succeeded...**

***“Nobody should start a large project. You need to start with a small, almost trivial project, and you should not realistically expect it to get large. If you do, you’ll over design...or be scared away. Don’t think about a big picture or a fancy design. Focus on solving an immediate need. Don’t expect things to happen in short time frames. I’ve been doing LINUX for 13 years and I expect to be doing it for quite some time still. If I had expected to something big, I’d never have started.”***

***--- Linus Torvalds (Linux Times, June 2004)***

***“Don’t think of short-term application development. Think like a city planner. Use the email system as an example: it’s an ever-changing collection of pieces of software and servers and accounts and other things...and it’s remarkably stable...it’s a ‘natural’ system that didn’t get defined or designed by anybody. Rather, it morphed it’s way into existence. When you start building a city, you think about where you’ll be in 5-10 years.”***

***-- Lisa Dusseault, WebDAV Chair***

# Questions?

