Why is C4I Software so Hard to Develop?

Why is C4I software

- Horrendous to design & develop?
- Ghastly to test & deploy?
- Monstrously expensive
- Hideously complex?
- Agonizing to upgrade & maintain?

June 2007
Dr. Lee Whitt
Technical Fellow
Northrop Grumman
Why is Investment Software so Hard to Develop?

Why is Investment software
- Often wrong
- Unable to predict future prices
- Unable to understand markets
- ....

June 2007

Dr. Lee Whitt
Technical Fellow
Northrop Grumman
Is C4I Harder than Picking Stocks?

- Which is harder:
  - Developing financial apps for investors
  - Developing C4I apps for warfighters

- Requirements and drivers
  - Greed drives investors
  - Everything drives the battlespace
Key Questions about building C4I Software

- What is easy? Hard?
- How long is the design phase?
- What is the importance of:
  - Architecture? technology? Standards? Governance? etc...
  - Test & certification? Deployment? Life-cycle support? etc...
  - Interoperability?
- What is the role of business logic in C4I?
  - What part is easy? Hard?
  - How long is the design phase? Development phase? Test phase?

- Business logic makes applications useful -
  Business logic transforms data into information & knowledge
Why might we think C4I software is easy?

- **Advanced technology is magical**
  - Vendor marketing is compelling
  - New technology means easier solutions
  - Early deployments are successful

- **Software is like hardware**
  - Hardware is rapidly refreshed
  - Hardware is very reliable
  - Hardware is easy to use

- **Governance breeds success**
  - Focus on process improvement
  - Engage stakeholders, engineers, users

Old technology is the problem!

Software should be like this!

It’s not that complicated!
C4I Information Integration Framework
Stock Information Integration Framework

Investor

Abstraction Layers
Awareness
Significance
Information relevant to mission goals
Level 3

Information
Object & Event Relationships
(Data in context)
Level 2

Data
Observables
Level 0/1

High

INFEERENCE LEVEL

Fusion

Process Refine (Lvl 4)

Context: Models, Pedigrees

Dynamic Re-plan

Resource management

Information Needs
Methods

Tasks
Plans

Inference

Markets
News
Money
Analysts
Politics

Low
Why is C4I Software Hard to Develop?

Answer: *Business Logic* !!

C4I business logic is the DNA of C4I systems

- **Rule-sets must be complete and consistent**
  - Rules must be specific (nature abhors a vacuum and program logic abhors a generality)

- **Rule-sets must be mission-specific and must accommodate myriad conditions within the mission context**
  - Difficult to define context boundaries
  - Difficult to define the variables (e.g., constraints, priorities, relationships)

- **Edge cases, ambiguity, & uncertainty must be addressed**
  - Rule-sets must balance type 1 and type 2 errors
Why is C4I Software Hard to Develop? (Continued)

C4I business logic defines C4I capabilities

- **Rule-sets operate within a mission context**
  - Transforming data into knowledge requires context
  - Legacy systems excel at maintaining context - *loosely coupled systems don’t*
  - System optimization (e.g., parallelization) can wreak havoc on rule-sets

- **Deep interoperability requires consistent & managed rule-sets**
  - Embedded rule-sets in legacy systems have evolved over many years
  - Interoperability has been achieved through common software
  - SOA designs don’t effectively address:
    - Legacy rule-sets
    - Context dependencies
    - Interoperability across new rule-sets
Why is C4I Software Hard to Develop? (Continued)

C4I business logic is the dark matter of C4I systems

- Rule-sets defy pattern analysis
  - Architectural patterns
    - DoDAF OV, SV, TV, ...
  - Software patterns
    - MVC, CRUD, ETL, ...
  - Process patterns
    - CMMI, Six Sigma, ...
  - GUI patterns
    - Style guides, ...
  - Rule-set patterns: ????
Recommendations

- Governance must expand to address business logic (and associated test plans)
- System design should accommodate technical diversity
  - SOA is not optimal or desirable for everything
- Promote interoperability through re-engineering legacy systems
  - Don’t just bolt on a few web services
- Leverage ‘rules engines’ as modular components
  - Provide web services to answer the “why?” question
  - Promote rule synchronization
- Engage industry groups focused on codifying rule languages and models
  - OMG, W3C, BR Community and on-line BR Journal
Closing Comments

- To much focus on:
  - SOA Technology - not enough on the business logic
  - SOA Technology interoperability (e.g., XML schema & semantics) – not enough on business logic interoperability

- SOA is much harder than client/server & n-tier
  - SOA will leverage legacy systems for the next decade (for as long as the business logic remains solely in legacy systems)

- SOA is an important technology
  ..........but SOA is not a solution

Software without business logic is like a child without adult supervision