The Massive Data Problem

Challenges and Strategies

Jim Thomas
President/Chief Technology Office
MetaTech Consulting, Inc.

Agenda

- Introduction to MetaTech Consulting
- Characterization of Massive Data
- Massive Data Generators
- Challenges that come with Massive Data
- Architectural Issues Concepts
- Design Factors
- Implementation Considerations
- Integration & Migration
- Summary and Closing Remarks

MetaTech Consulting, Inc.

- Information Management Systems architecture and engineering services.
- Special emphasis is given to the challenges presented by the *Massive Data Problem*.
- Principally support the DoD and other Federal agencies – mostly within the Intelligence Community.
- All fulltime consultants hold TS/SCI clearances with full-scope polygraph
- System Engineering & Technical Assstance (SETA)
  - Applied Technology
  - Innovations
- Http://metatechconsulting.com

Preface

- Architecture v. Engineering
- Vision
  - How much do you have?
  - Where will your data needs me in the future?

Characterization of Massive Data

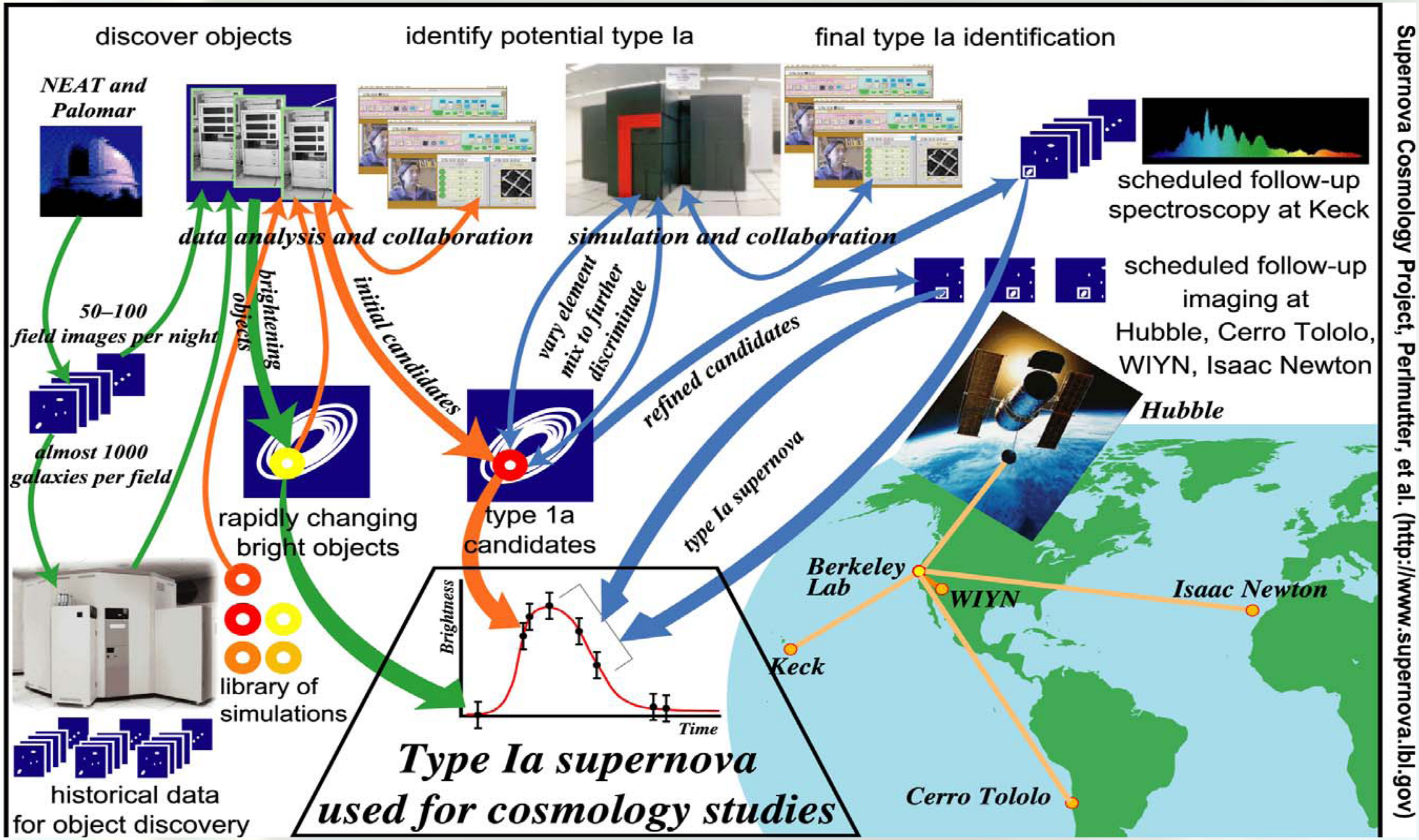
- How *big* is it?
  - Terabyte (10^{12} bytes of data)
    - Telecommunications Call Detail Warehouse
    - National Retail Point Sale Data
  - Petabyte (10^{15} bytes of data)
    - Text and Images Product Description
  - Exabyte (10^{18} bytes of data)
    - National Medical Insurance Records
  - Zettabyte (10^{21} bytes of data)
    - Spatial and Terrestrial Data
    - Video and Audio Archive Data
  - Yottabyte (10^{24} bytes of data)
    - Moore's Law: Database size in 2050

Carino, Kauffman, & Kostamaa (2000)

Massive Data Generators

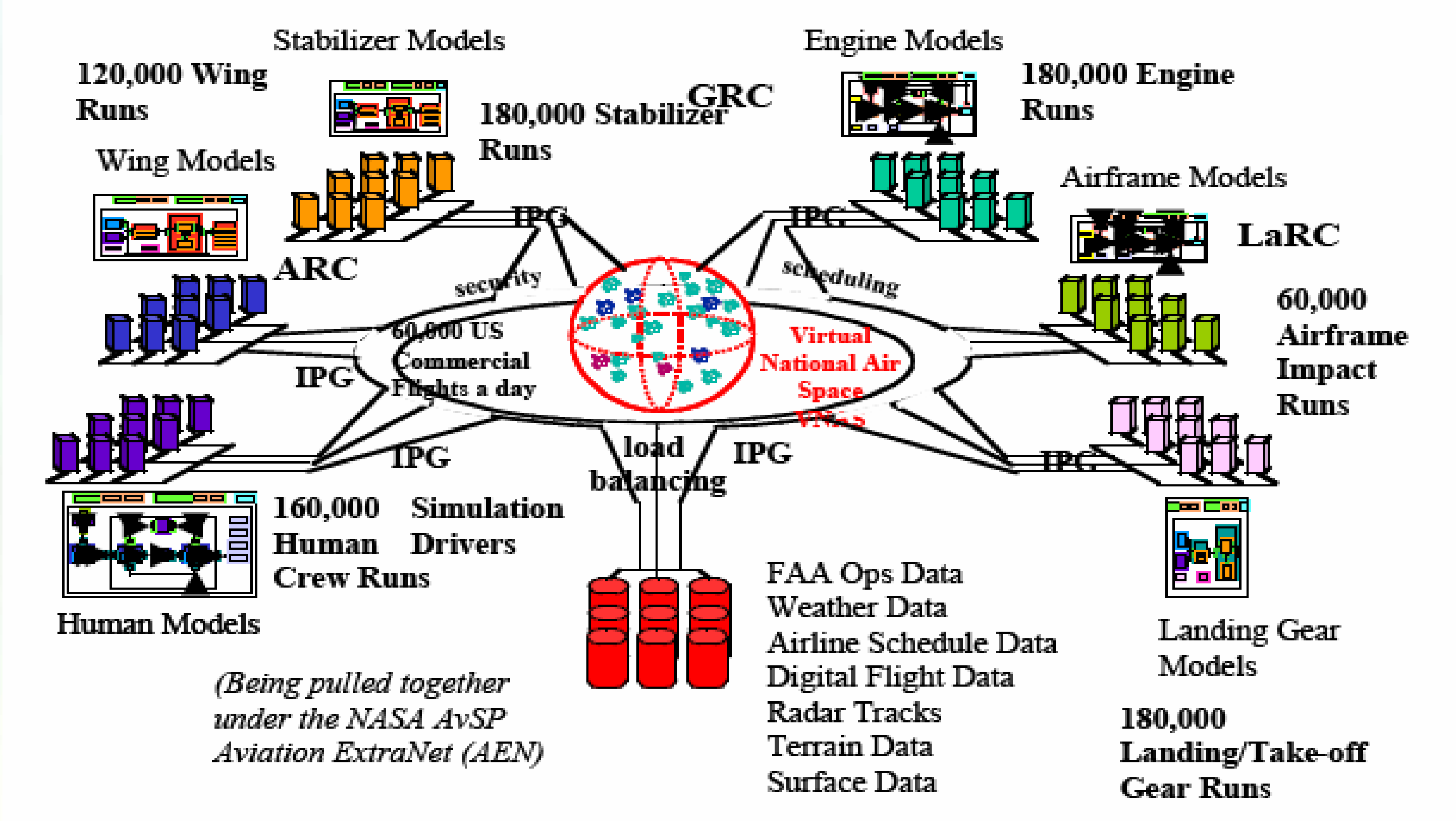
- Deep archives
  - Telecommunications industry
    - Years of call records
    - Example: Sprint IP Backbone – 600 gigabyte packet trace data per day
- Wideband sensors
  - Space borne platforms
    - photographic sensors
    - Meteorological
  - Particle physics (http://www.griphyn.org)
  - Cosmology (http://www.supernova.lbl.gov/)
- Integrated networks of disparate sensors
  - Virtual Air Space Simulation Environment (http://ic-www.arc.nasa.gov/publications/pdf/2000-0204.pdf)

DOE's Supernova Cosmology Program



http://www.lbl.gov/supernova/

Virtual National Air Space Simulation Environment



http://ic-www.arc.nasa.gov/publications/pdf/2000-0204.pdf

Challenges

- Quantity of data
- Rate of ingest
- High availability demand (24 x 7)
- No “window” for ingest (or backup)
  - Simultaneous and continuous ingest and access
- Streaming data
  - Can’t stage data during ingest
- Disparate data models
- High security demands

Architectural Activities

- Enterprise-wide strategy
  - Context, Scope, etc.
  - Architect *globally* engineer *locally*
- Codify the enterprise
  - Model everything….but….
    - Only to the detail necessary
    - Avoid “analysis paralysis”
- Decompose the problem
  - More manageable pieces
  - Solvable with available technologies…mostly

Architecture Standards

- ISO/IEC 12207.0 – 1996: Standard for Information Technology - Software life cycle processes
- IEEE 1471: Standard for Architecture Description (2001)
  - Specifies normative requirements for architecture
  - Specifies *architectural views*
    - Functionality
    - Performance
    - Security, and
    - Feasibility.

Supportive Architectural Concepts

- Distributedness
  - Distributed != Federated
  - Databasing
  - File Systems
- Layered Storage
  - This is not in the context of Hierarchical Storage System
  - Data and metadata have distinct management schemes

Design Factors

- Logically integrate data
  - Conceptual data model (Ontology)
  - Functional Data model
- Manage content indirectly
  - Through metadata

Implementation Considerations

- Warehousing
  - Its not just for data anymore…
  - Provide the infrastructure for the *Metadata Solution*
  - Utilize the file system to manage large content artifacts
  - Store a handle to the artifacts with other metadata in the warehouse (RDBMS).
- Storage Area Networks
  - Flat-file storage
  - Database storage
  - Distributed

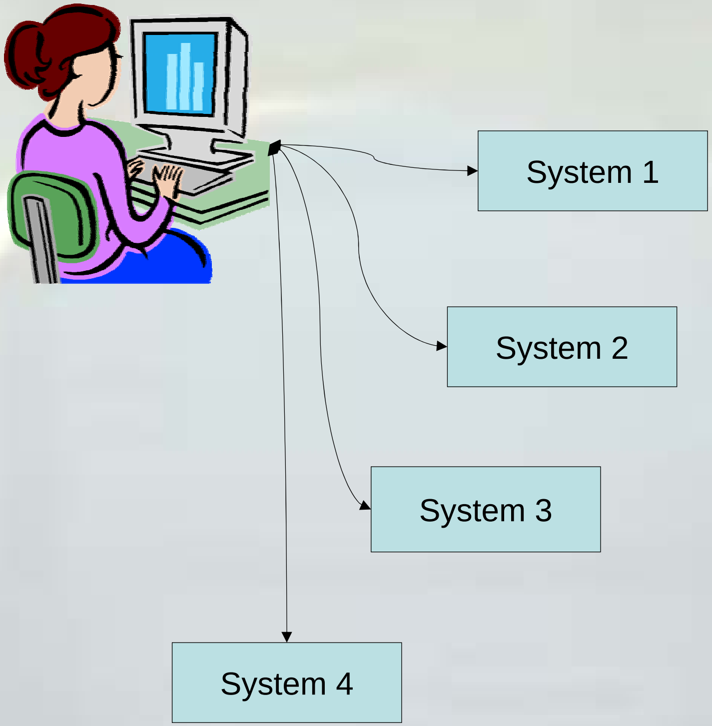
Levels of Integration

- Use Integration
- Programmatic Integration
- Interface Integration
- Component Integration

Use Integration

- User is the functional interface between systems
- Different tool for each system.
- Little or no data interchange.
- Necessary data conversions is achieved through tools or utilities.
- Not to be confused with data fusion. This form of integration may facilitate rudimentary fusion analysis, but that is not the sole driver.

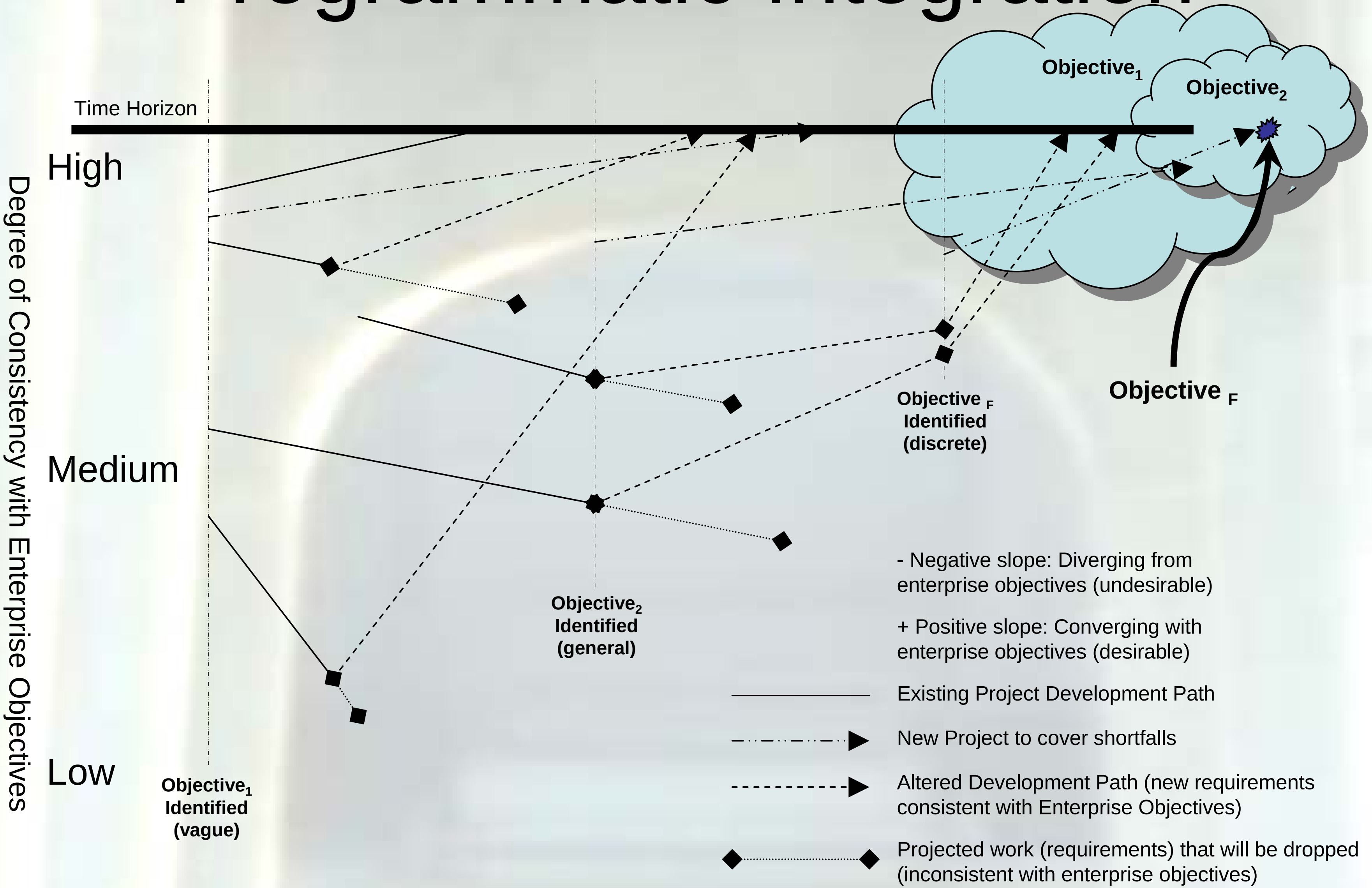
Use Integration



Programmatic Integration

- Project management teams are working to a single plan
- Common goals and objectives
- Common budget management

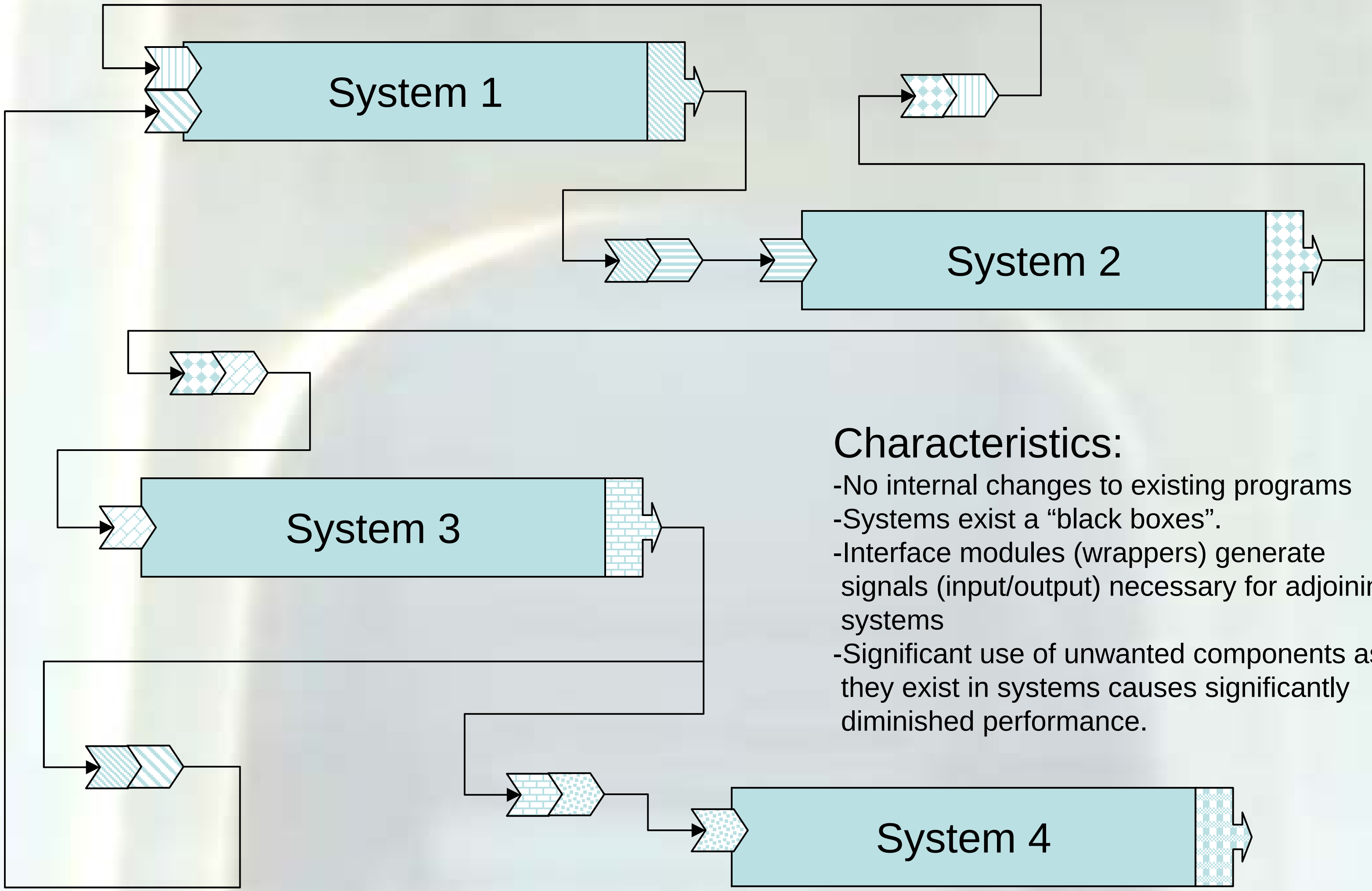
Programmatic Integration



Interface Integration

– Interoperability only through interface wrappers
– Common infrastructure services possible only through wrappers
  - Inefficient and expensive

Interface Integration



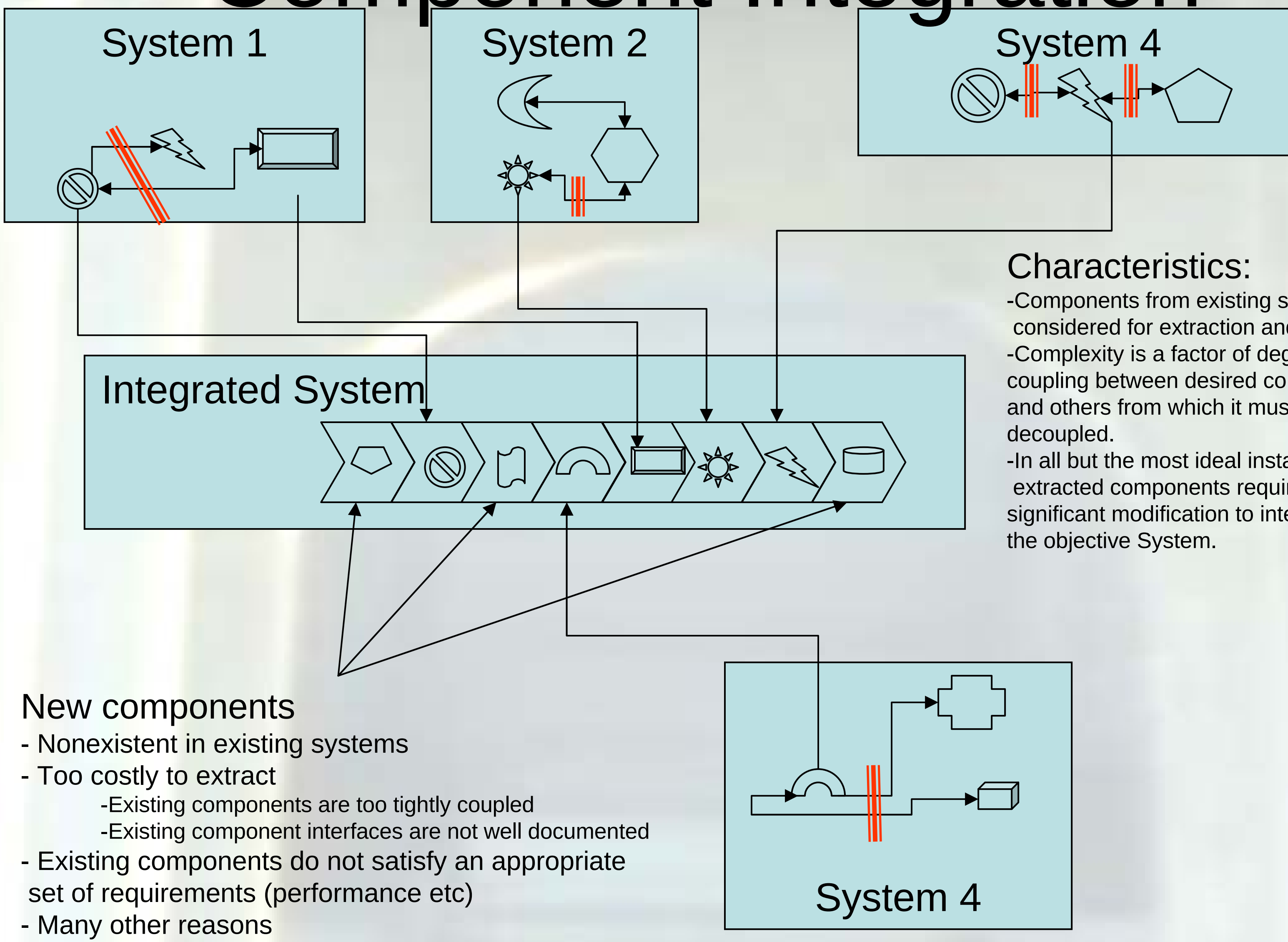
Characteristics:

- No internal changes to existing programs
- Systems exist a "black boxes".
- Interface modules (wrappers) generate signals (input/output) necessary for adjoining systems
- Significant use of unwanted components as they exist in systems causes significantly diminished performance.

Component Integration

- Full data interoperability
- Code reuse
- Full plug-and-play
- Recapitalization of development efforts

Component Integration



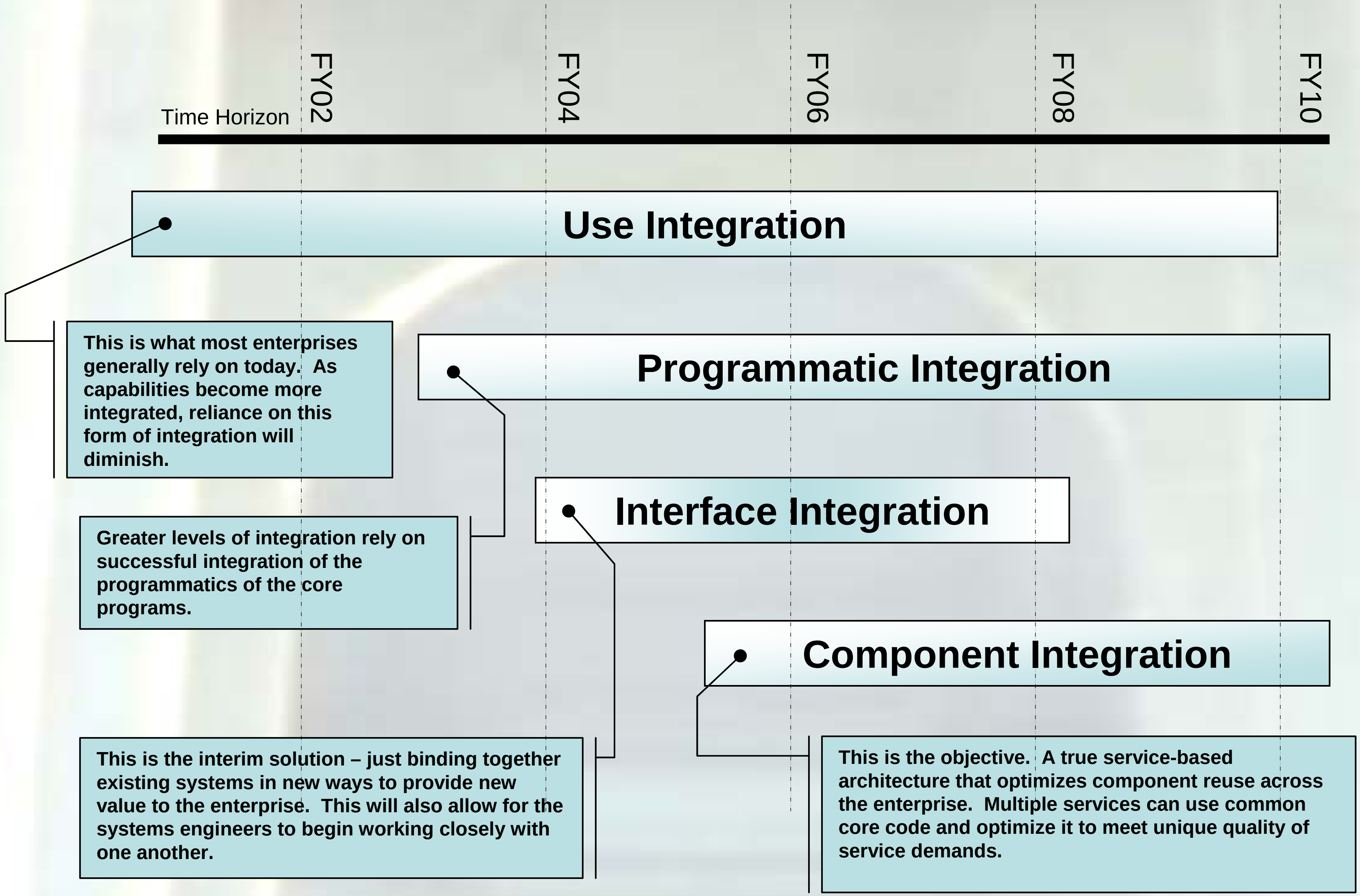
Characteristics:

- Components from existing systems are considered for extraction and reuse.
- Complexity is a factor of degree of coupling between desired components and others from which it must be decoupled.
- In all but the most ideal instances, the extracted components require significant modification to interface with the objective System.

New components

- Nonexistent in existing systems
- Too costly to extract
  - Existing components are too tightly coupled
  - Existing component interfaces are not well documented
- Existing components do not satisfy an appropriate set of requirements (performance etc)
- Many other reasons

Migration


Time Horizon
FY02
FY04
FY06
FY08
FY10
Use Integration
This is what most enterprises generally rely on today. As capabilities become more integrated, reliance on this form of integration will diminish.
Programmatic Integration
Greater levels of integration rely on successful integration of the programmatics of the core programs.
Interface Integration
This is the interim solution – just binding together existing systems in new ways to provide new value to the enterprise. This will also allow for the systems engineers to begin working closely with one another.
Component Integration
This is the objective. A true service-based architecture that optimizes component reuse across the enterprise. Multiple services can use common core code and optimize it to meet unique quality of service demands.

MetaTech Consulting, Inc

Jim Thomas
President/Chief Technology Officer
202.368.2177
jim.thomas@ieee.org

Biliography

- Bardinia, J., McDermott, W. J., Follen, G. J., Blaser, T. M., Pavlik, W. R., Zhang, D., & Liu, X. (2000). *Integrated Airplane Health Management System*. Retrieved on 10 October, 2003 from http://ic-www.arc.nasa.gov/publications/pdf/2000-0204.pdf
- Carino, Kaufmann, & Kostamaa (2000). *Are you ready for yottabytes? Storehouse Federated and object/relational solution.* Retrieved on January 10, 2004 from http://siteseer.nj.nec.com/carino00are.html
- **Johnston, W. E. (2002). *Computational and Data Grids in Large-Scale Science and Engineering*. Future Generation Computer Systems, 18 (8), pp. 1085-1100.**