A research consumer's view of data integration

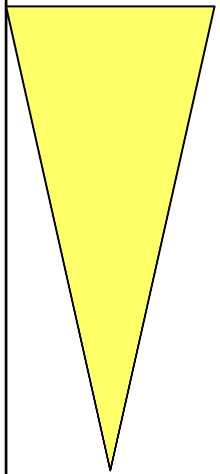
Arnon Rosenthal
The MITRE Corporation
Bedford MA

MITRE

Outline

- Where we stand
- Some principles for giving research greater impact
- Ontology vs. database integration styles – strengths and weaknesses
- Open problems + sore points receiving little attention

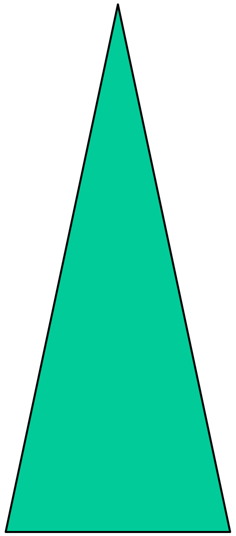
Researchers are *too* loosely coupled to practical data sharing



Research history

- Virtual db with distributed query
- Data warehouse
- Formatted messages

- Lightweight scripts (pairwise)
- Integrated packages (e.g., SAP)
- Cobbling together an exchange i/f (including data standards)



Industrial usage

Data sharing strategy

(from enterprise transformation advocates)

- Service oriented architecture
  - SOA doesn't solve your data mess, it reveals it [Gartner]
  - Lacks general ad hoc query, update, evolution help, replication support, declarative constraints, …
- Communities of interest will create shared specs ("stds")
  - Community = ?? (organization? mission thread?)
    - System implements separate interface for each mission thread to which it belongs?
  - All members will "adopt"? Implement?
  - Need clear definitions, metrics, tools to support such processes
    - E.g., what tradeoffs for community size, scope, cohesion, ...

MITRE

Data sharing real practice, in US tactical systems

- Warehouses for logistics and other “back office”
- Little federated query
- Each new application does the “integration” it needs

- Independent tactical systems, communicating by agreed message types (since 1970s, moving to XML)
  - Wrapper interfaces are plug compatible
  - But wrappers are inflexible, created manually, and redundant: (SOA limitations apply – query, update, …)

- Agree on tiny interface constituents, e.g., “what/when/where” or mashup on geo-location
  - Good return on investment, but doesn’t scale up

  Tiger teams reduce from months to weeks / days for ~15 elements

MITRE

Pervasive problems -- 1

- Lack of metrics. You can't contract for what you can't measure
  - How much useful data sharing have we enabled?
  - Agility (even w.r.t. schema changes)

- Skills are in short supply
  - Organizations can't employ a technique requiring skill in logic, ontologies or XQuery
  - Integration engineers need skills in all stages

  Move toward single-skill tasks – a "supply chain" with tools weaving the pieces together

MITRE

Pervasive problems -- 2

- **Low penetration for data technologies**
  - Database paradigm (schema, query) vs. business objects
  - Data integration tools – need cheaper, easier start up
- **Incentives are not explicit artifacts in models**
  - Systems are unusable, because someone doesn't care enough to do what's asked
  - Repositories are either unfunded, or not forced to be useful
  - Create tools that use m'data to benefit the m'data provider
  - Manage incentives, as another architectural view
- Families of systems, e.g., a system is deployed to many places, and extended separately in each

MITRE

Outline

- Where we stand
- Some principles for giving research greater impact
- Ontology vs. database integration styles – strengths and weaknesses
- Open problems + sore points

Ideas for greater research impact

- Automate *something*, don’t just assist
- Downstream work transitions more easily
  - Evolution is a killer app for “downstream”
- Help even when the requirements can’t be met automatically

MITRE

Solve *something* -- don't just "assist"

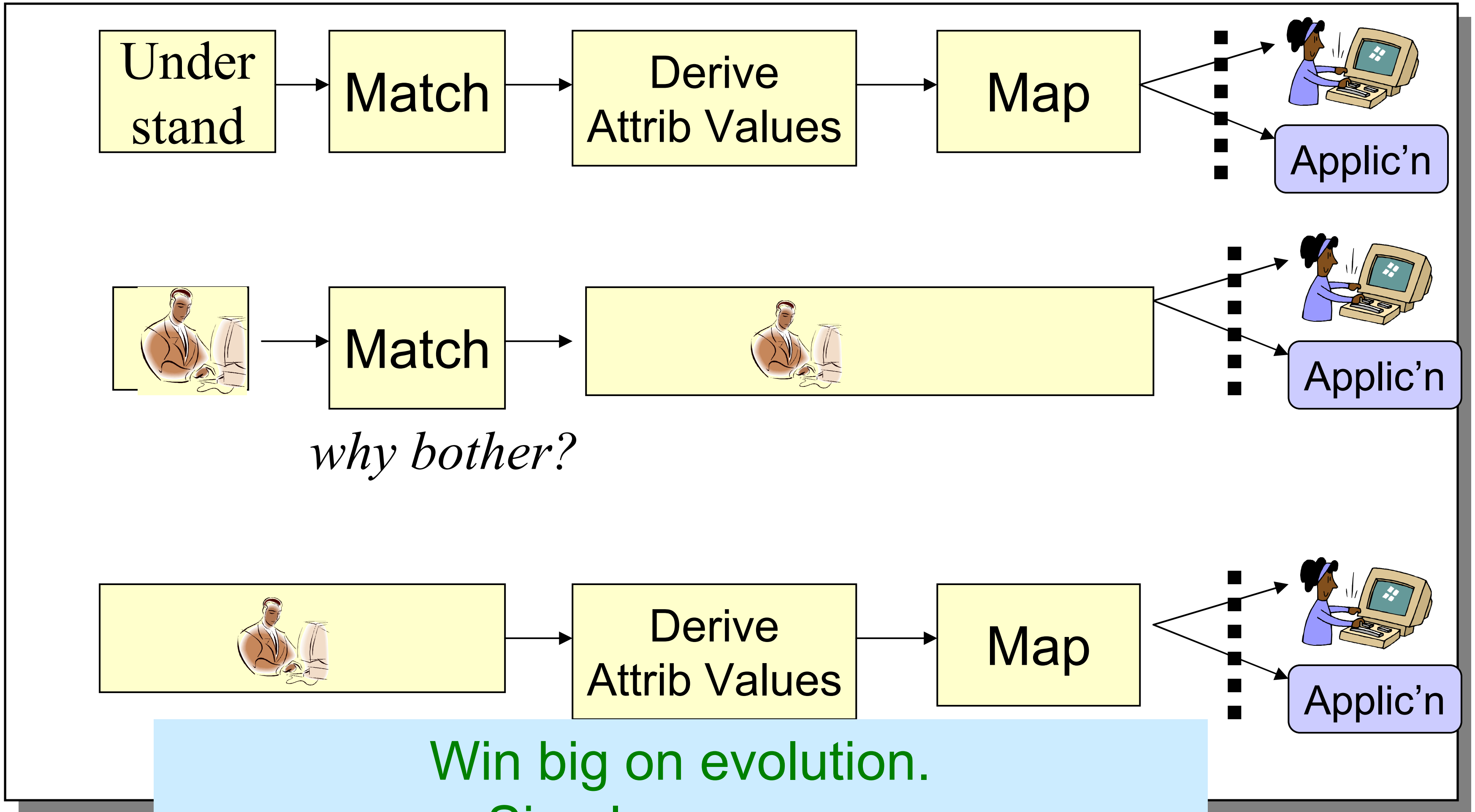
A fully automated modular piece has greater potential for a big win (e.g., GUI-builder)

- Deploy new instances without deploying people
- When cost goes to zero, new uses will be found
- Modularity forces a clean problem definition, allows indep. Improvement
- "Assist" lets humans help when tools don't work together
- You may need human review, e.g., precise Match

- Candidates for full automation
  - Best efforts (e.g., for discovery)
  - Evolution
  - Cluster of variants on a theme

MITRE

Transfer downstream first



Win big on evolution.
Simpler process
Need less skill the first time

So, what should we solve?

- "Match first", with mapping left manual?
  - ≈40% saving *there to end. No revolution!*
  - *Too many skills.* Mapping will require domain expert + programmer, causing delays, misunderstandings
- Automate "map" ⇒ ≈99% saving *there to end*
  - Evolve: Change just upstream of automation ⇒ immediately runs. *Something is fast.*
  - Simpler process: Less switching between humans and tools
  - Less need for skills
- Reduce number of people in the value chain (esp. with rare skills but not unique domain knowledge)

MITRE

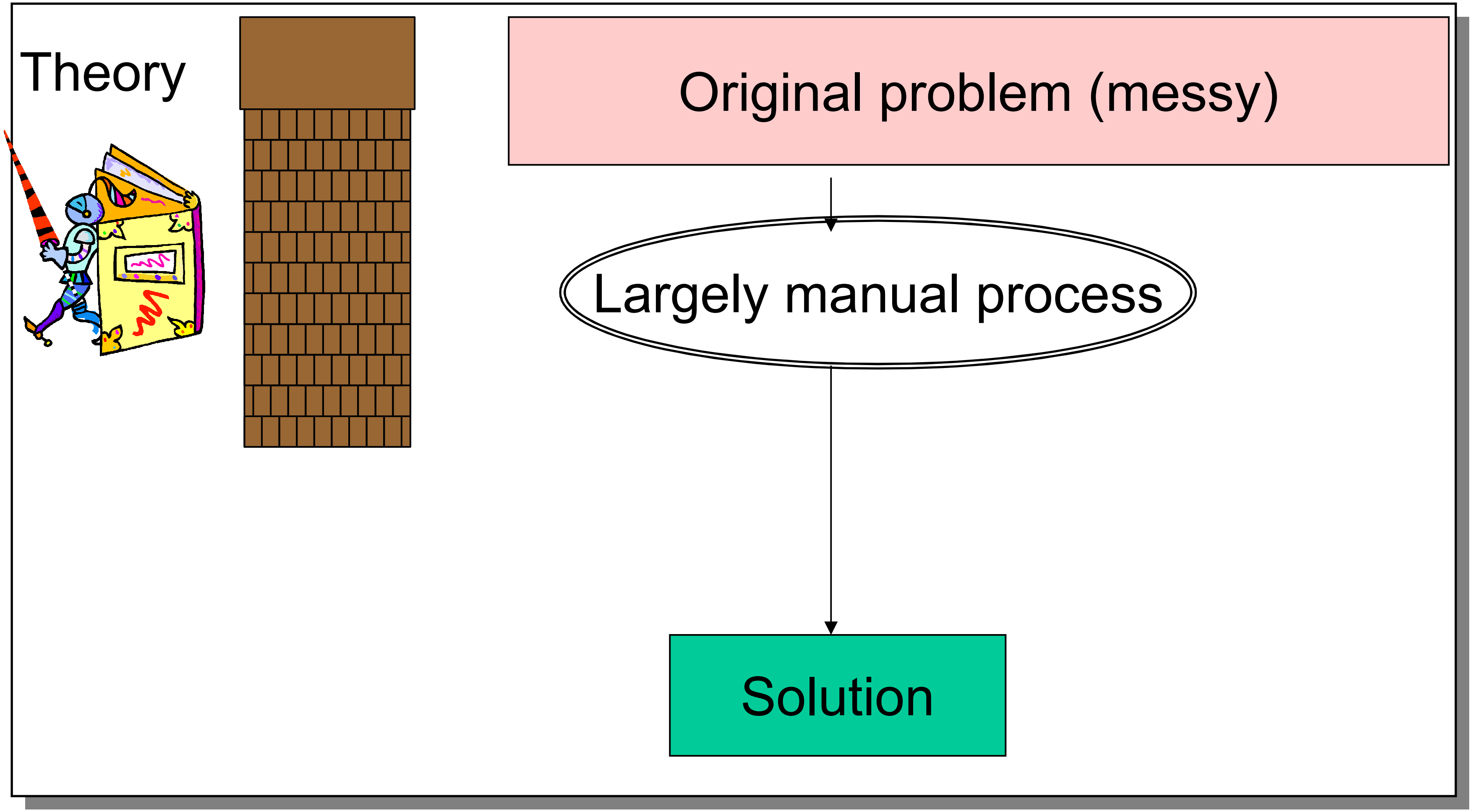
Scalability requisite: Avoid global assumptions

- Suppose we assume "X *always* holds", e.g.,
  - *All* participants employ enterprise standard data elements (or schemas)
  - All participants are willing to shift to use our new process framework
  - *All* Views are Select/Project/Join
  - *All* constraints are Key, Foreign Key, or unary
- If X fails, the theory becomes logically empty
- Do I then buy a separate software system?

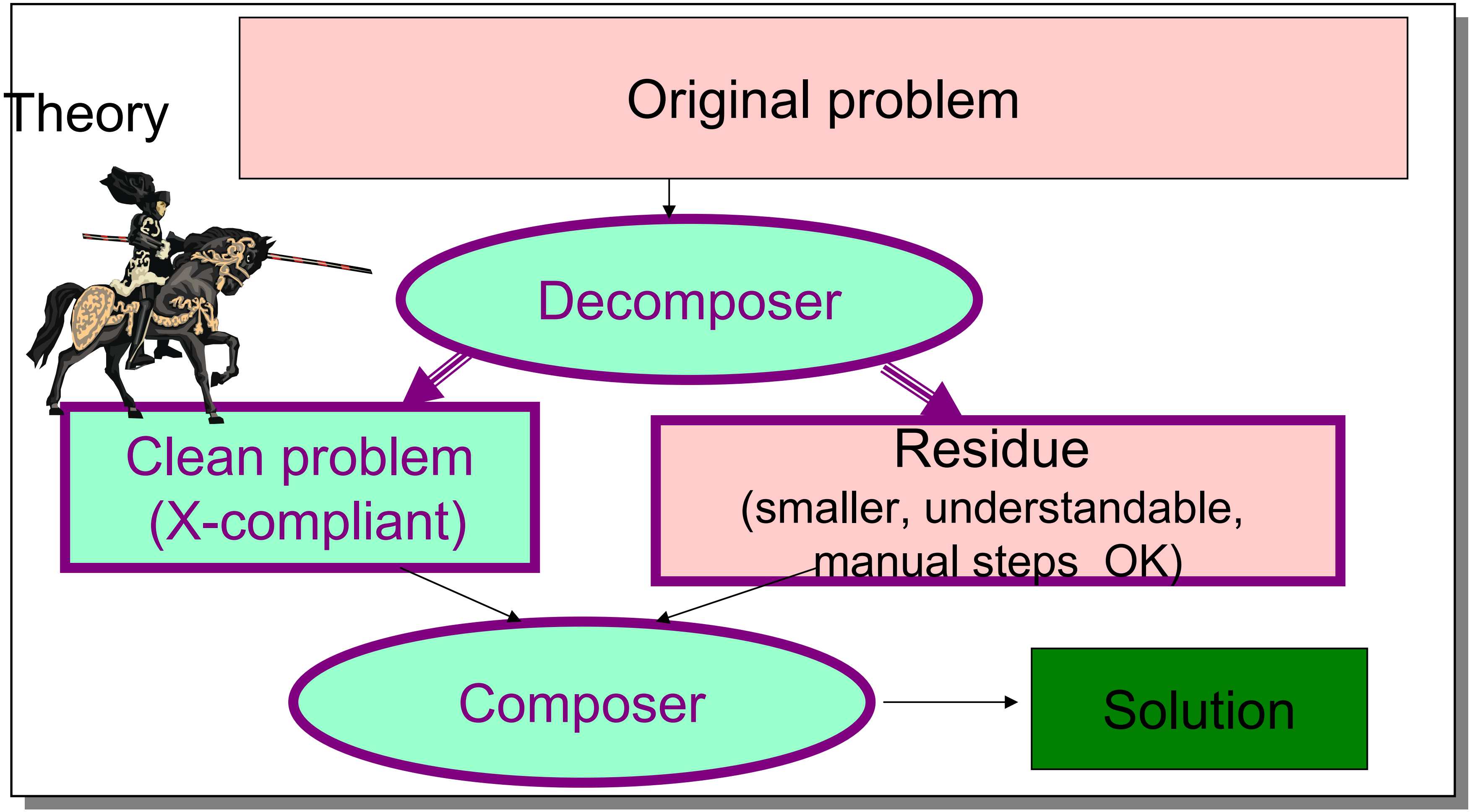
You can't control a large enterprise (legacy, mergers, future needs, + partners)

MITRE

What we get with a restricted theory



More robust:
Researcher gives formal decomposition



More robust:
Researcher gives formal decomposition

- Algorithm to extract the X-compliant portion
- Solve the clean part
- Create a clean residue problem
  - Solvable (possibly with human tasks)
  - Understood by your target integrators (don't "reduce" to logic)
  - Smaller than the original (preferably in same form, or simpler)
- Algorithm to recompose the pieces

MITRE

Example: Define a nice subproblem that you can solve

- *Tractable*: "nice" mappings to an unconstrained schema
- Intractable: many constraints on target schema, e.g., keys, null not allowed

- Null not allowed : If sources have insufficient info, they can't fill in the target
- Key constraint: Logic won't resolve conflicting source assertions
- Inclusion: Formally OK, but who has permission to insert? Non-key? [Keller 86]

MITRE

Approaches – 1
"Bad constraints" go into residue

Data cleansing is a good source of inspiration – abstract and extend its insights

- Split
  - Exchange to an tractable target

    e.g., target relation schemes, maximal target constraints implied by sources,

    *and **understandable by users***
  - Exchange from that target to desired results

- This residue problem seems
  - **Much simpler** (no structural differences)
  - **Understandable by intended users**

MITRE

Approaches – 2
"Bad data" go into residue

- Partition **data** automatically: easy $\cup$ hard
  maximize "easy"
  - Niche: situations where it's safe to exchange *part* of the data

- Violate key constraint
  - All tuples involved in violation are "hard"
- Violate inclusion (key? nonkey?)
  - Does *application* want to insert automatically?
- Violate value constraint
  - Need cleansing rules?
    Aircraft.Name alphanumeric, cleanse F-16 → F16

MITRE

Outline

- Where we stand
- Some principles for giving research greater impact
- Ontology vs. database integration styles –strengths and weaknesses
- Open problems and Sore points

MITRE

Compare typical(??) DB vs. AI approaches (1)

Formalisms to describe concepts & relationships

DB (Schema)

- Basic unit: relational or tree (XML) schema
  - Record is a good chunk for storage or display
- Describe a system or a physical message
  - Plug compatible, *without mediator*

AI (Ontology)

- Basic unit: atomic concept (object or property)
  - Small chunks ⇒ easy to relate & reuse
- Describe a *neutral* domain model (more community based, e.g., science)
  - Robust for multiple uses
  - **Mediator required**

MITRE

Compare typical AI vs. DB approaches (2)

DB

- Schemas make little use of IS-A, constraints
  - Developers add much of the semantics
- Relationships among ***sets***, via query language, or logic
  - TGDs are awkward, unfamiliar to users
  - View defns are big: hard to edit and partially reuse (e.g., ETL script as view)

AI

- More use of IS-A and constraints
  - Motivated by easier query formulation, reasoning
- Relationships among ***concepts***: "Usable_for"
  - Employs formalism similar to *within* ontology (supertype of IsA property)
  - Relationship pairs are small, easy to edit (and to reuse)

MITRE

Compare typical DB vs. AI approaches (3)

What models do we connect

(for semantic relationships? for data flows)

DB (Schema)

- Between systems
  - Instant gratification (often fund one app, not integration)
  - Differences in *real data* lead to improved definitions

AI (Ontology)

- Via neutral defns or structure (msgs)
  - Reuse is easier
  - "First create an ontology" inhibits sales
  - Do admins understand "foreign" or abstract defns, well enough for precise integration?

MITRE

Compare typical AI vs. DB approaches (4)

DB	AI
• Exchange semantics: precise, relevant, complex, justified? – Hard to learn or communicate – Works with whole tuples. Discards partial info if *some* field can't be decided (??)	• Exchange semantics: Whatever *my* engine infers !!! – How to separate the easy cases (no join) – How to explain the (in)signficance of the problem to ontologists and managers

MITRE

“Usable for” relationship

- Ontology formalism can be used to describe either systems or domain models
- *Usable_for* is directional, to relate concepts that are not the same
- Usable_for does not force inheritance – one system may have fewer attributes
  - A super-type of the IS-A property

Compare typical AI vs. DB approaches (5)

DB	AI
• Homegrown logic – Reason well about crucial integration constructs – Even simple Datalogs won't interoperate – Extensible	• OWL has a larger developer community – Extensible
• Execute popular query language in robust server – Efficient, parallel, **deployable**	• Execute by inference engine
• **Change mgt is *sometimes* careful**	• Change mgt tends to be ad hoc

MITRE

Outline

- Where we stand
- Some principles for giving research greater impact
- Ontology vs. database integration styles –strengths and weaknesses
- Open problems and Sore points

MITRE

Communicating with Muggles

- Our sponsors don't use the researchers' distinction of integration vs. exchange
- Info integration = (?)
  - Given a data need (target schema + constraints), populate it from available source(s) +
  - Given multiple sources, create a product that combines some of their data

MITRE

Data sharing >> Info Integration

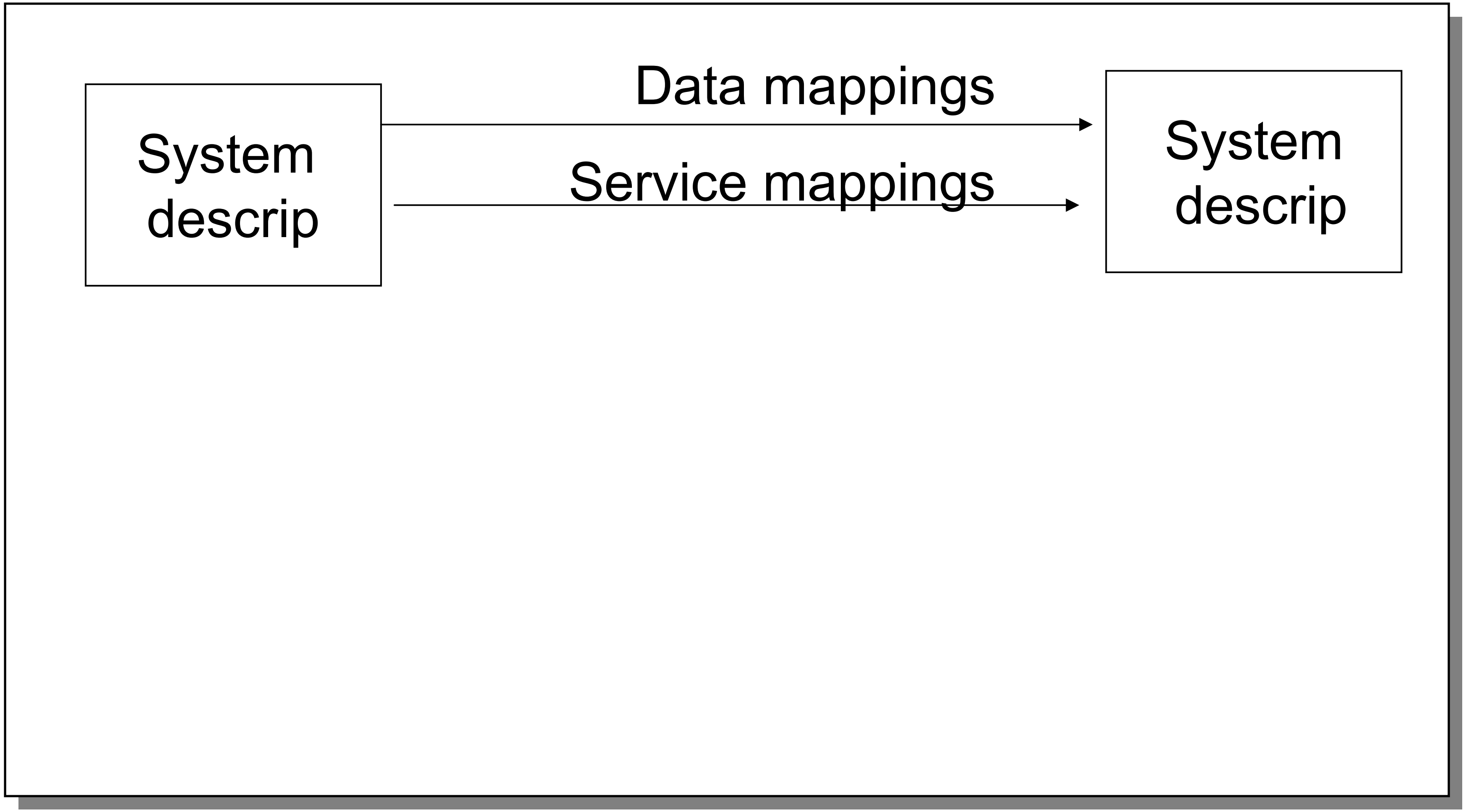
- Data sharing at MITRE brings in and social scientists and security folks
  - Nontechnical barriers (resistance, incentives, obligations)
    [My website has several papers -- Seligman, Swarup]
  - *Suitably* assure that no data leaks across boundaries – Retain flexibility

MITRE

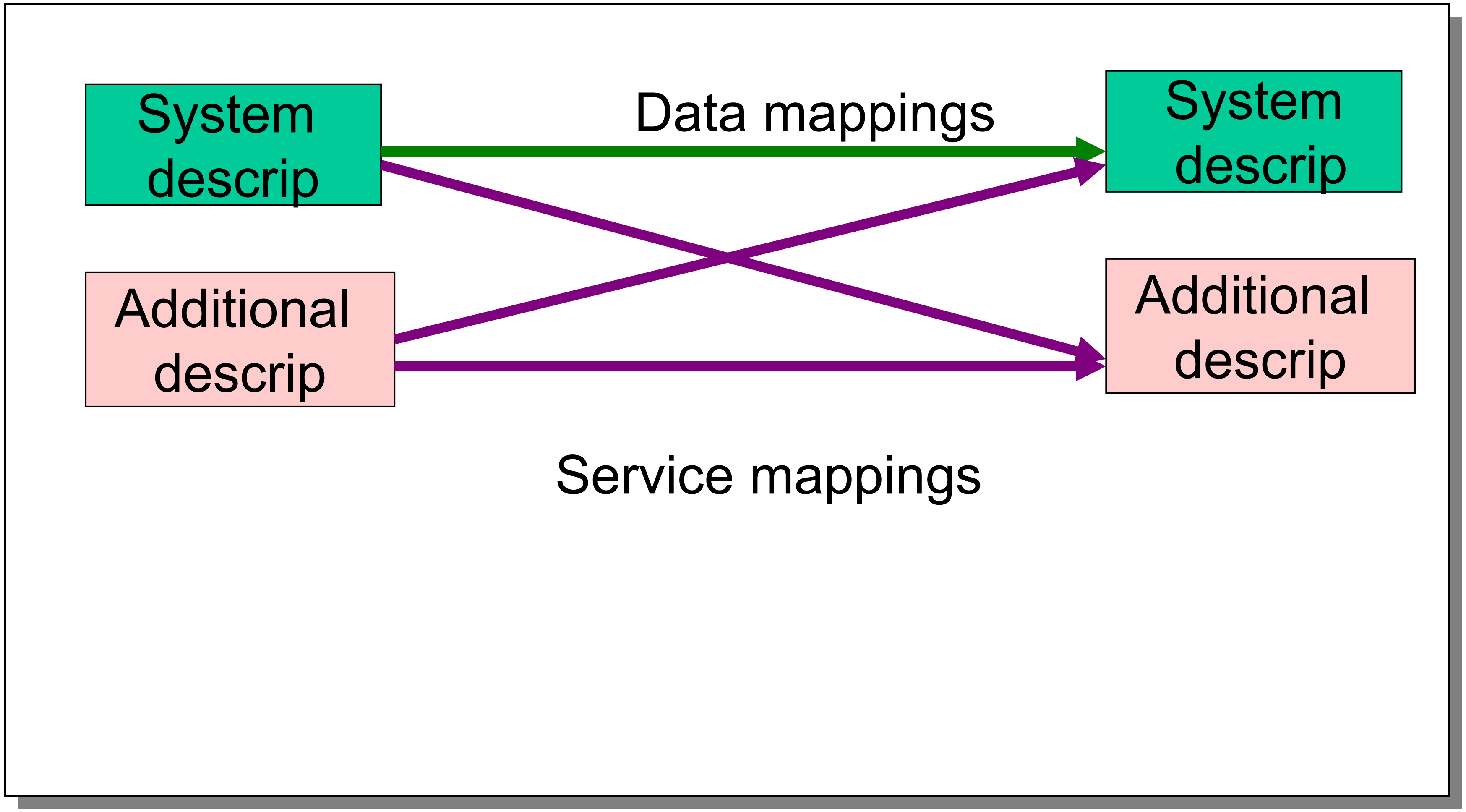
Open problems
(at least, for products)

- Integrate applications, at data, business object, and display tiers (i.e., multiple system views, simultaneously)
  - Long term goal: Capture more of system descriptions as data
  - Integration capability can be a big incentive
- Examples
  - Integrate data *and also* declaratively-defined displays [Ceri…., Raghu]
  - Data + Style sheet?
- Other complex types with multiple system views, e.g.,
  - Business process descriptions, …
  - Social networks

MITRE


System descrip
Data mappings
Service mappings
System descrip

“Design patterns” for progressing



More open problems

- Variations on a theme (e.g., the many extensions of a popular system)

MITRE

Smarter synthesis: Don't just integrate what fate has provided

- 95% of our research examines integration of existing data or interfaces
- 80% (?) of tactical data sharing involves
  - Hammering out shared specifications (msg formats, view interfaces)
    - Otherwise, source may not capture the info you need (e.g., landing time = ? leave runway)
  - Reusing the same specification in interfaces to multiple systems (e.g., "what/when/where" XML fragment)

Mismatch

MITRE

Community-created specifications

- Proper direction: Groups (communities?) create specifications
- Someone decides to incentivize their use
  - How to measure? How to manage? What mechanisms?
- Researchers can start by clarifying definitions and abstract problems
  - Define and justify metrics and best practices
  - Create operators to synthesize new products from a variety of agreed ones

  Plenty of applicable theory, but not synthesized for this purpose

MITRE

Exploit uncoordinated progress

Niche: Many integration efforts, with limited coordination

- A popular system will be deployed many places
- Individuals will extend each one differently
- Examples
  - Spreadsheet definitions are widely copied, and extended by recipients
  - A major intelligence DB is set up in separate countries, sharing most of the schema, some data
- When is it safe to share their extensions, without harming applications?
  - May not have the framework to manage views to shield
  - XML is tougher, e.g., when add nodes within paths [Mork]

MITRE

Frameworks are needed

- How will we combine all the emerging ideas? (and those from folks outside the room)
  - Today, each idea needs to be reimplemented in each tool
  - Poor business model for small players

  Result(?): General purpose integration environments are costly (Db import may not be, e.g., SQL Server)
- Need (open source) frameworks
  - Macro: “Network effects” in integration drives consolidation. Big ones need one to combine the software they acquired.
  - Micro: Frameworks for execution, testing, tuning can improve learning steps (e.g., alignment)

Learning, for each component problem

- Reuse collaborative experience
  - When several integration efforts occur in parallel, someone may have answered your question already
  - Many more sophisticated cases

  Show stopper: Our sponsors rarely have suitable repositories

- Learn by analogy
  - Learn about data sources (e.g., recognize units and formats)
  - Data conversions

    F-16 → F16; F-15 → F15
  - Entity matching and data cleaning rules

  Many users can do instances, few can specify rules

  [Miller et. al., for outerjoins etc.]

MITRE

Backup

MITRE

Multi-$Billion data integration industry, uses little research*

- E.g., ETL, configuration, memorandums of understanding
- * Exceptions
  - Federated query
  - IBM info integration (from CLIO)?
  - ADO.Net entity framework?

MITRE

DARPA may join the party

- Many folks from the AI side
- More for nonstandard applications, perhaps to support non-IT folks