Enough Already! How Data-Modeling Tools and Notations Compromise Information Quality—And What You Can Do About It.

ABSTRACT------------------------------

Information quality critically depends on consensus about the semantics of business information. Consequently, organizations formally model their information requirements, which are distinct from “system requirements.” Notations accommodating information requirements must a) exclude technological detail, b) include business detail, and c) be able to arrange information requirements along a heretofore ignored axis: “Business Significance.”

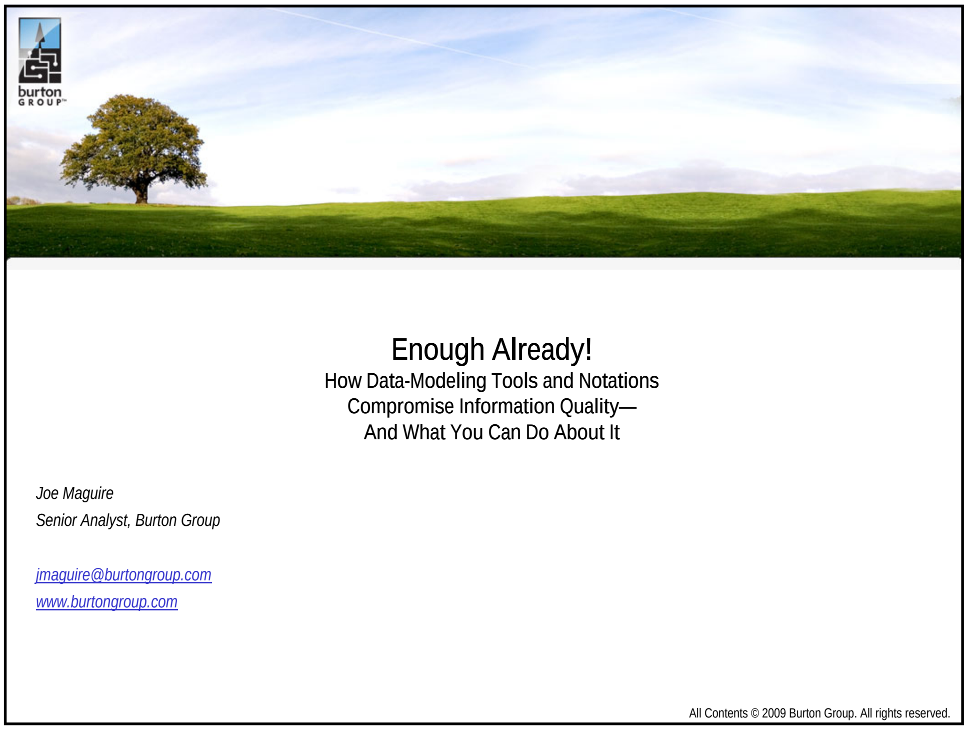
Business significance is important because two requirements A and B might look syntactically and structurally identical, but requirement A could be motivated by an invariant truth about the information, whereas requirement B could be motivated by something transient, such as current corporate policy. Such distinctions should influence the design of any system that will be expected to maintain information quality in a dynamic business environment.

This presentation describes—with real-world sample models—why requirements analysts should consider business significance, why existing notations (including NIAM-based methods) fall short, and how working analysts can word around these notational shortcomings.

BIOGRAPHY------------------------------

Joe Maguire
Senior Analyst
Burton Group

Joe Maguire is Senior Analyst at Burton Group specializing in data-modeling techniques and tools. During his 26 years in the software industry, he has worked in product development (for Digital, Lotus, Microsoft, and Bachman Information Systems) and has consulted for small startups and Fortune-100 companies in a wide range of industries including software R&D, pharmaceuticals, networking and telephony, mass-storage devices, publishing, and environmental engineering. His books—including *Mastering Data Modeling: A User-Driven Approach* (Addison-Wesley, 2000)—have been reviewed favorably by a wide range of publications including The *Mathematica Journal*, *Science News*, *The Data Access Newsletter* (TDAN.com), The Boston Sunday Globe, and National Public Radio.

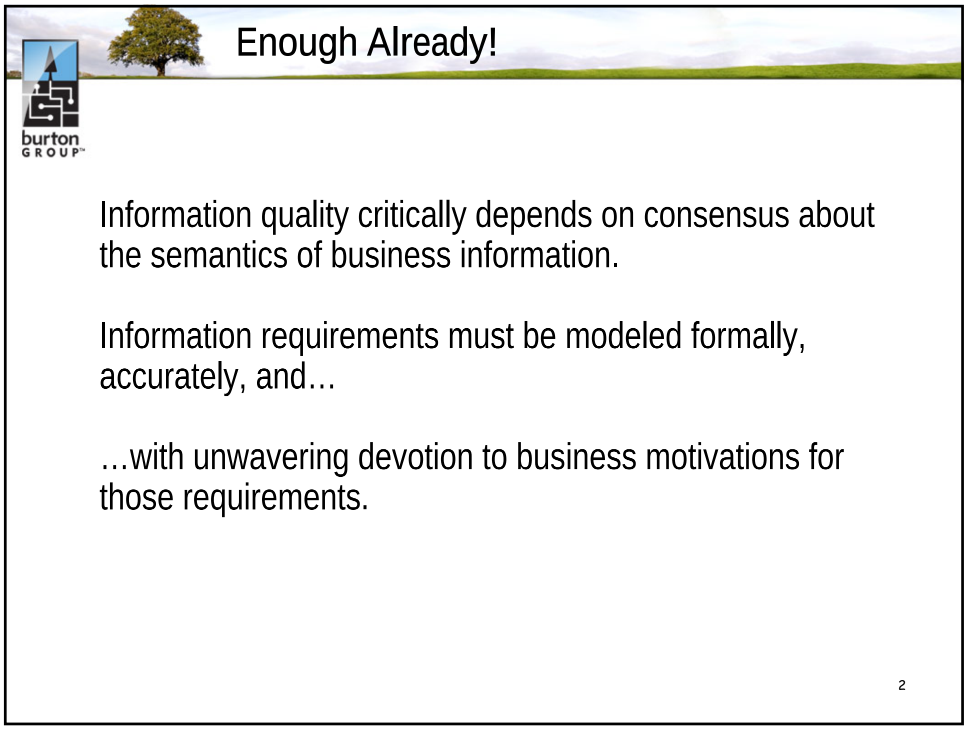


Enough Already!

How Data-Modeling Tools and Notations
Compromise Information Quality—
And What You Can Do About It

Joe Maguire
Senior Analyst, Burton Group

jmaguire@burtongroup.com
www.burtongroup.com



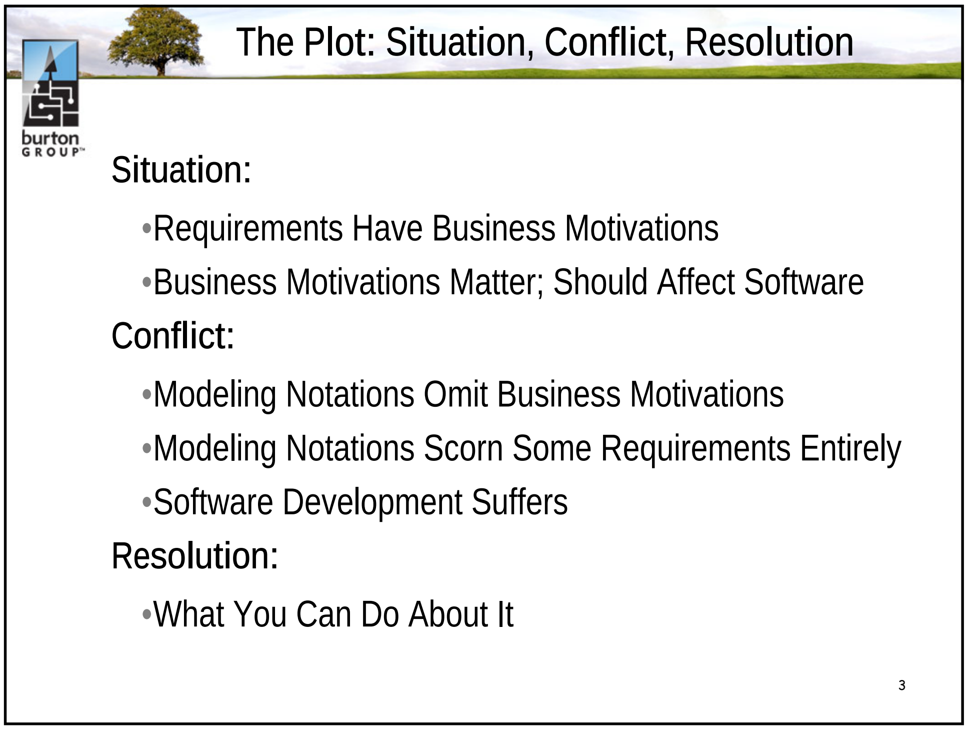
Enough Already!

Information quality critically depends on consensus about the semantics of business information.

Information requirements must be modeled formally, accurately, and…

…with unwavering devotion to business motivations for those requirements.

2



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

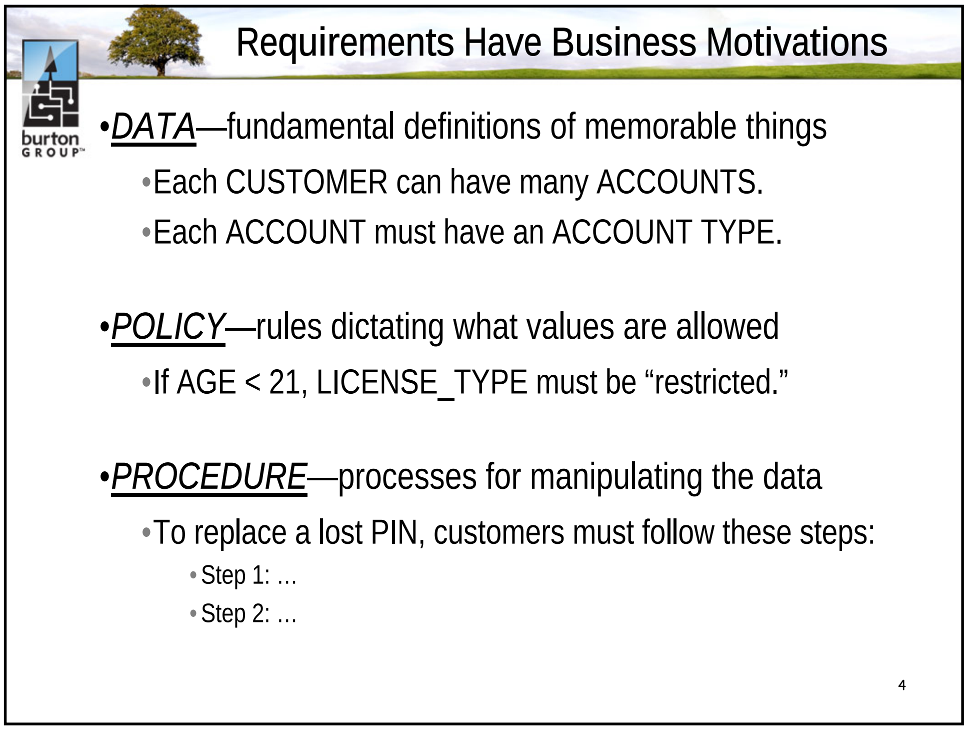
Conflict:

- Modeling Notations Omit Business Motivations
- Modeling Notations Scorn Some Requirements Entirely
- Software Development Suffers

Resolution:

- What You Can Do About It

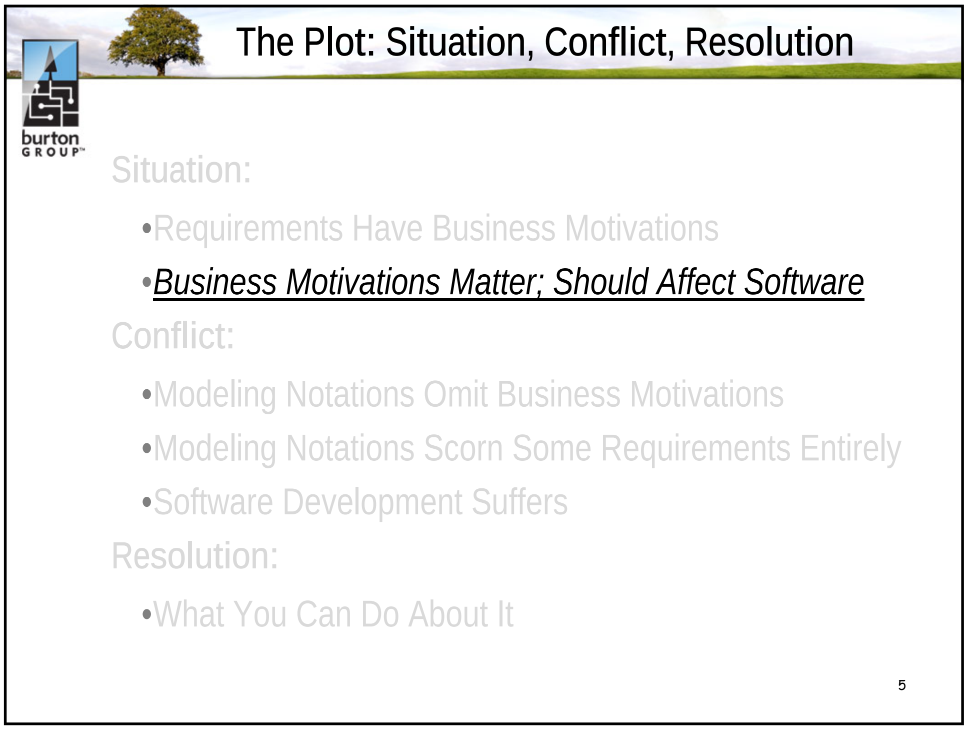
3



Requirements Have Business Motivations

- ***DATA***—fundamental definitions of memorable things
  - Each CUSTOMER can have many ACCOUNTS.
  - Each ACCOUNT must have an ACCOUNT TYPE.
- ***POLICY***—rules dictating what values are allowed
  - If AGE < 21, LICENSE_TYPE must be “restricted.”
- ***PROCEDURE***—processes for manipulating the data
  - To replace a lost PIN, customers must follow these steps:
    - Step 1: …
    - Step 2: …

4



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- *Business Motivations Matter; Should Affect Software*

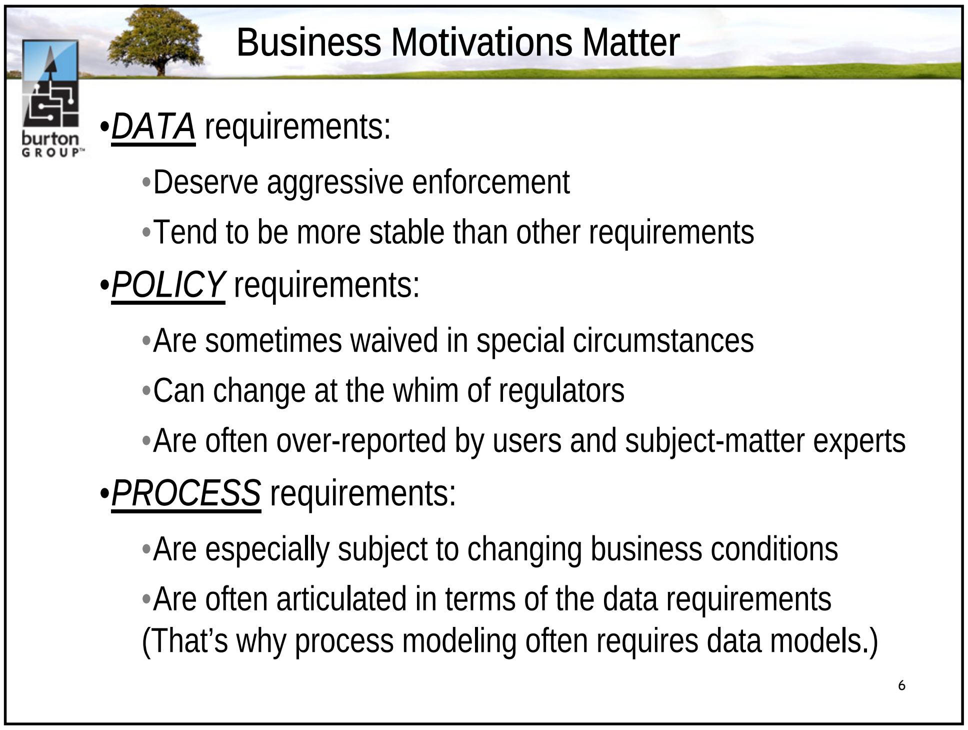
Conflict:

- Modeling Notations Omit Business Motivations
- Modeling Notations Scorn Some Requirements Entirely
- Software Development Suffers

Resolution:

- What You Can Do About It

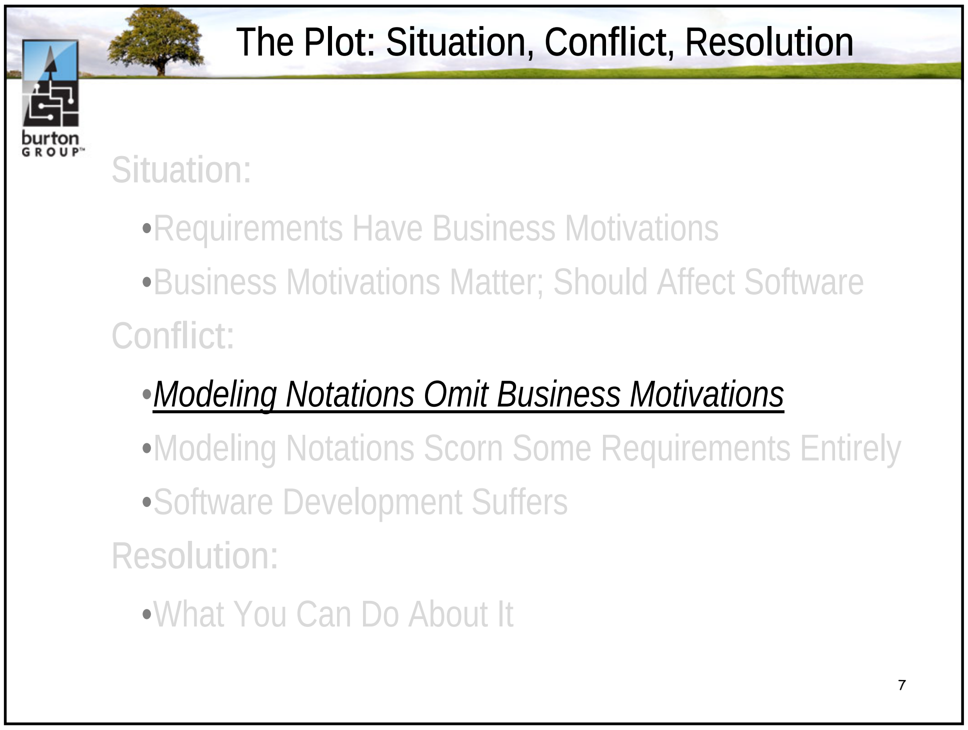
5



Business Motivations Matter

- *DATA* requirements:
  - Deserve aggressive enforcement
  - Tend to be more stable than other requirements
- *POLICY* requirements:
  - Are sometimes waived in special circumstances
  - Can change at the whim of regulators
  - Are often over-reported by users and subject-matter experts
- *PROCESS* requirements:
  - Are especially subject to changing business conditions
  - Are often articulated in terms of the data requirements (That's why process modeling often requires data models.)

6



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

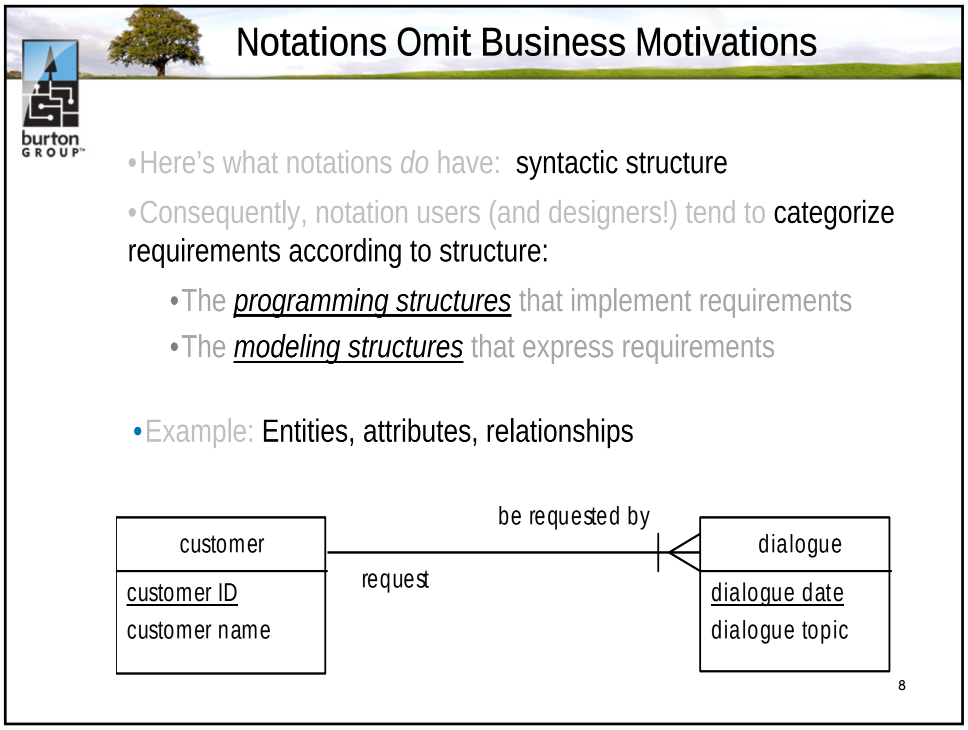
Conflict:

- ***Modeling Notations Omit Business Motivations***
- Modeling Notations Scorn Some Requirements Entirely
- Software Development Suffers

Resolution:

- What You Can Do About It

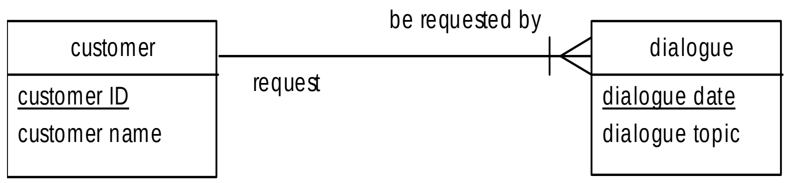
7



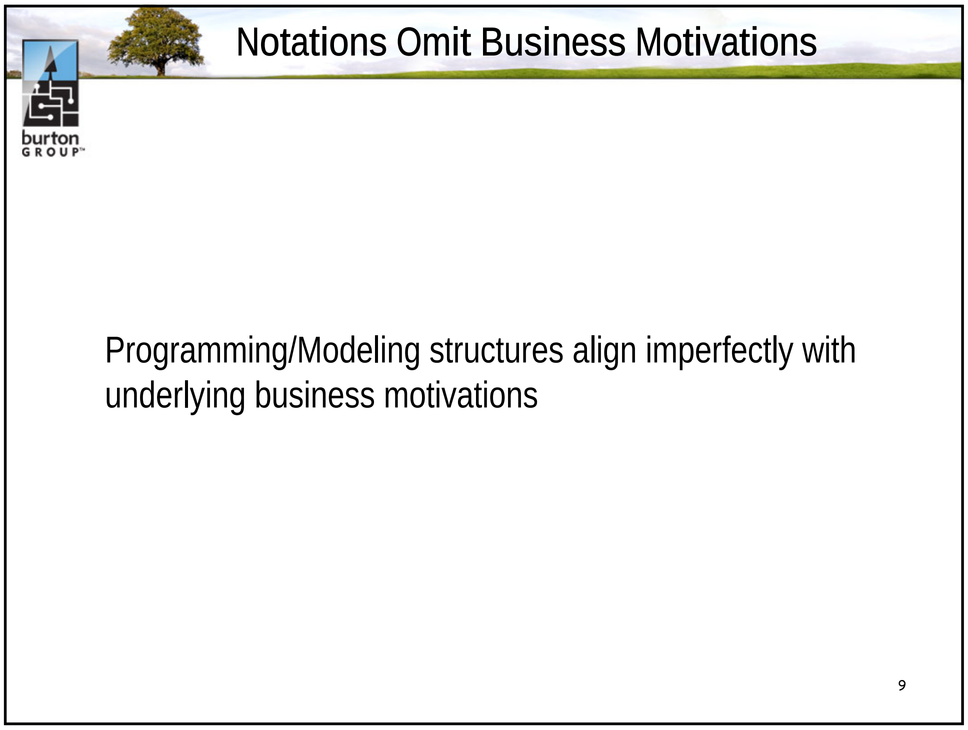
Notations Omit Business Motivations

- Here's what notations *do* have: syntactic structure
- Consequently, notation users (and designers!) tend to categorize requirements according to structure:
  - The ***programming structures*** that implement requirements
  - The ***modeling structures*** that express requirements

- Example: Entities, attributes, relationships



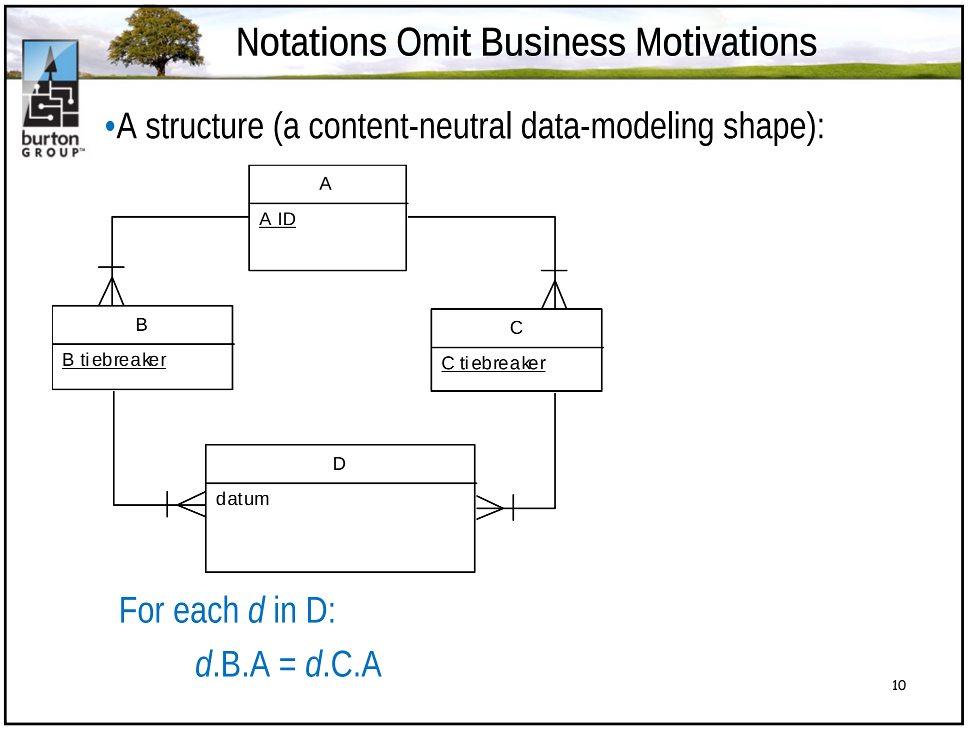
8



Notations Omit Business Motivations

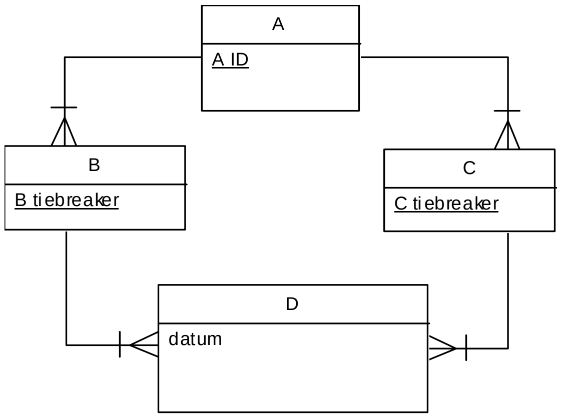
Programming/Modeling structures align imperfectly with underlying business motivations

9



Notations Omit Business Motivations

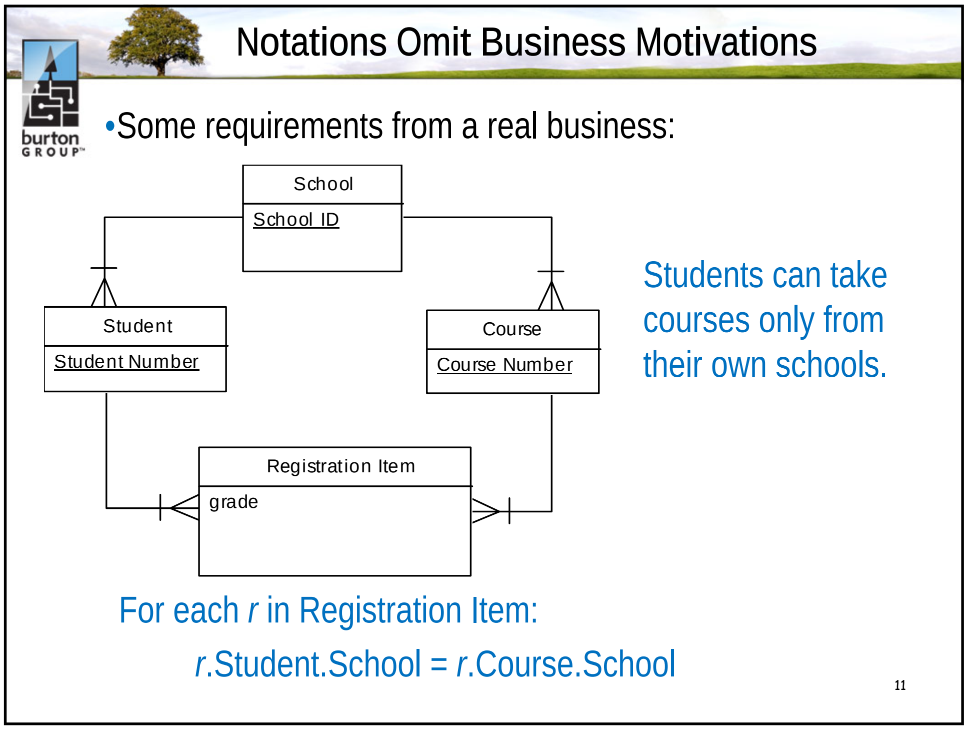
•A structure (a content-neutral data-modeling shape):



For each *d* in D:

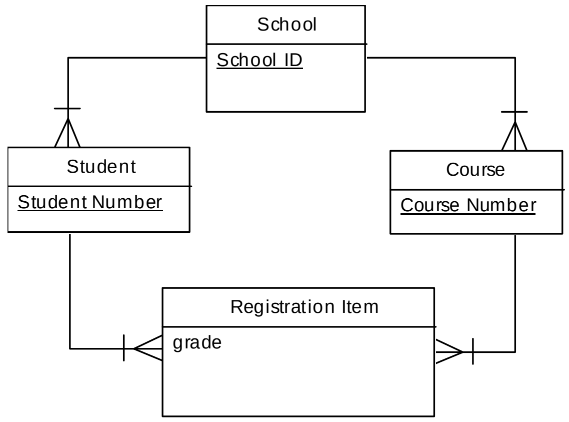
d.B.A = *d*.C.A

10



Notations Omit Business Motivations

•Some requirements from a real business:

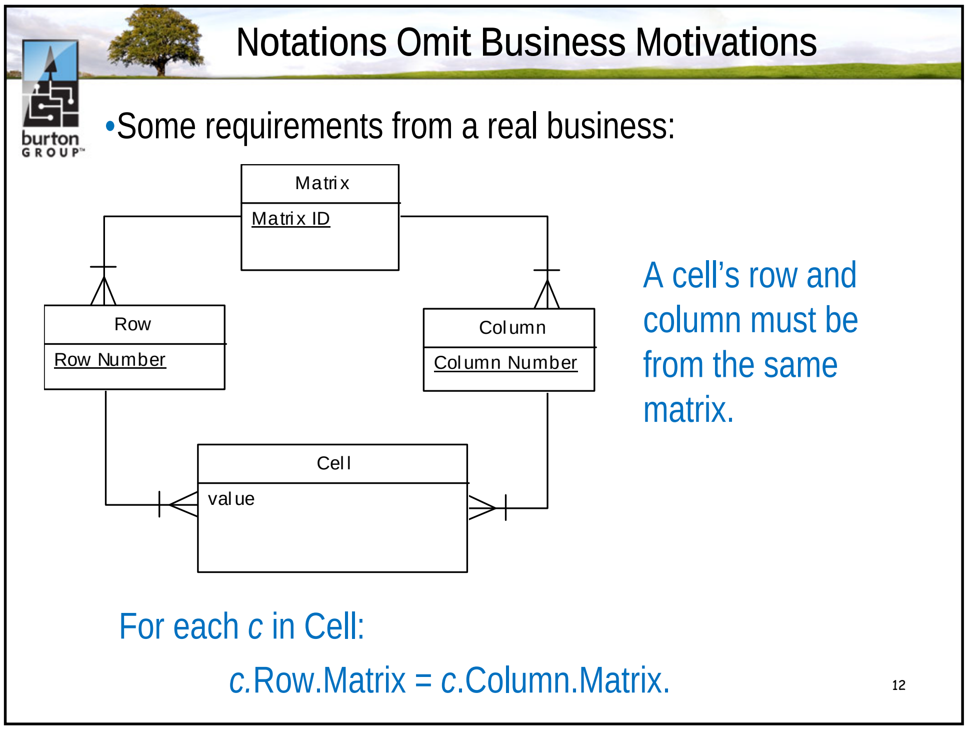


Students can take courses only from their own schools.

For each *r* in Registration Item:

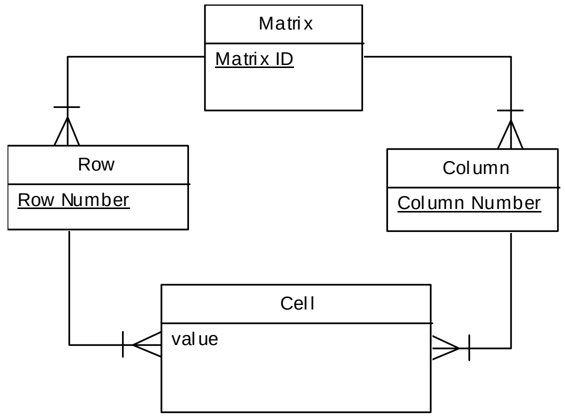
r.Student.School = *r*.Course.School

11



Notations Omit Business Motivations

•Some requirements from a real business:

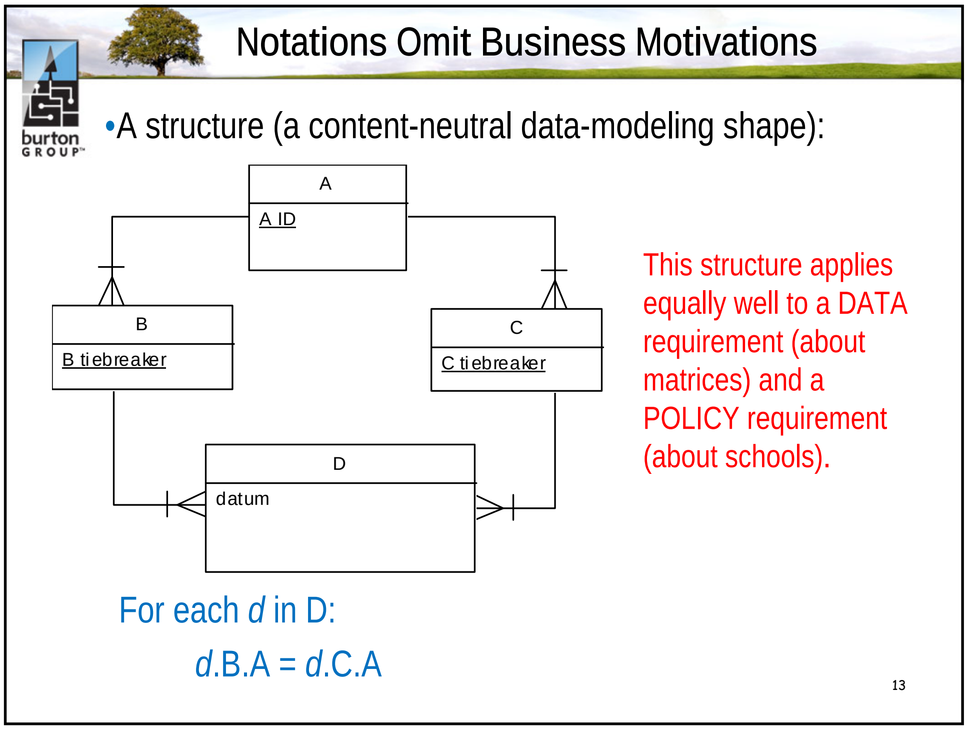


A cell’s row and column must be from the same matrix.

For each *c* in Cell:

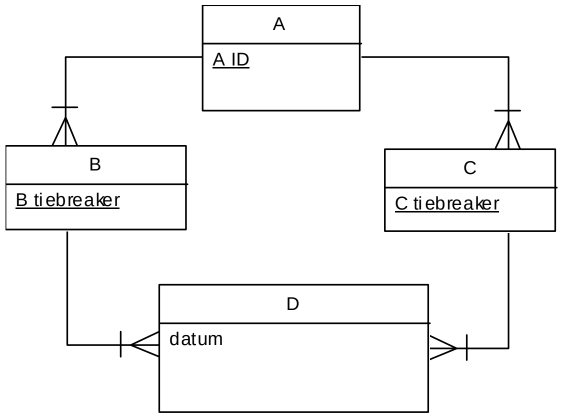
c.Row.Matrix = *c*.Column.Matrix.

12



Notations Omit Business Motivations

•A structure (a content-neutral data-modeling shape):

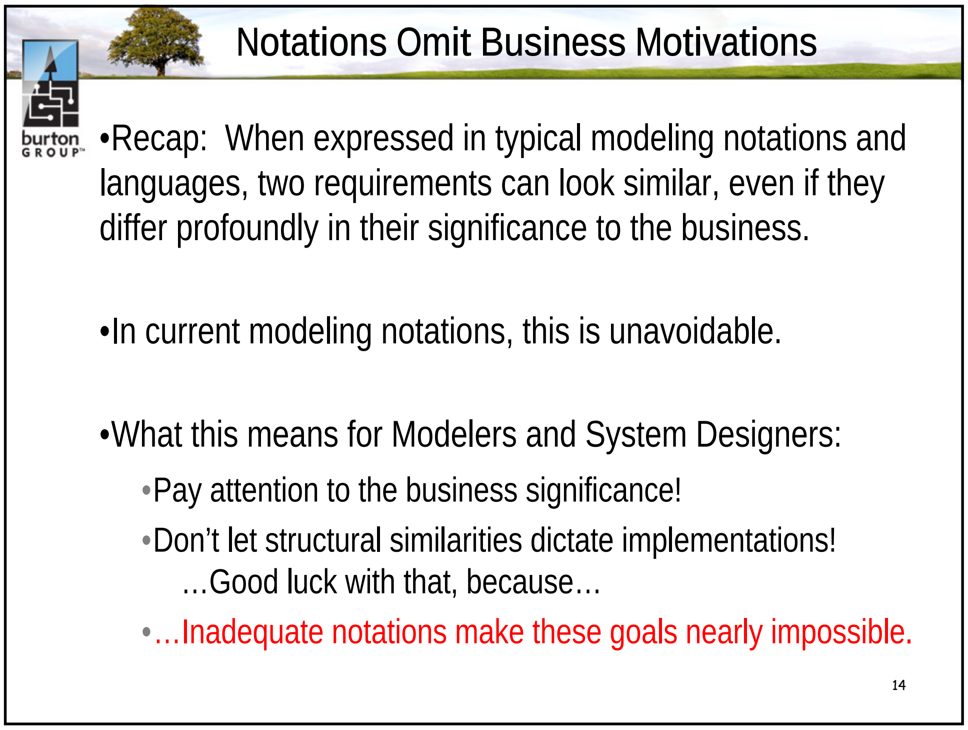


This structure applies equally well to a DATA requirement (about matrices) and a POLICY requirement (about schools).

For each *d* in D:

d.B.A = *d*.C.A

13



Notations Omit Business Motivations

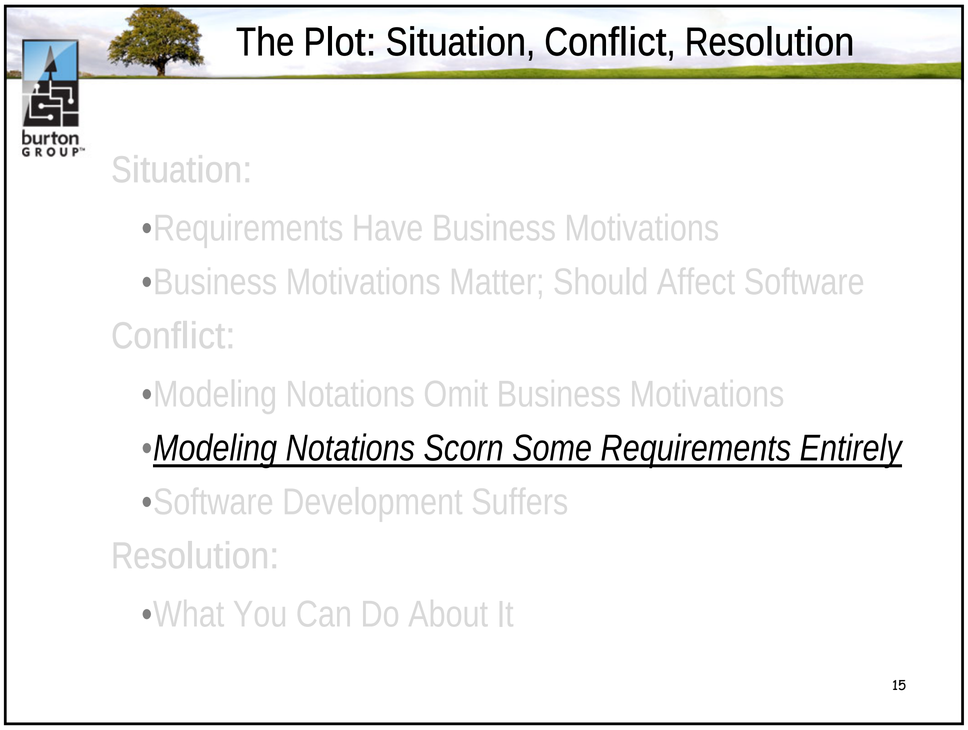
•Recap: When expressed in typical modeling notations and languages, two requirements can look similar, even if they differ profoundly in their significance to the business.

•In current modeling notations, this is unavoidable.

•What this means for Modelers and System Designers:

- •Pay attention to the business significance!
- •Don't let structural similarities dictate implementations!
  …Good luck with that, because…
- •…Inadequate notations make these goals nearly impossible.

14



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

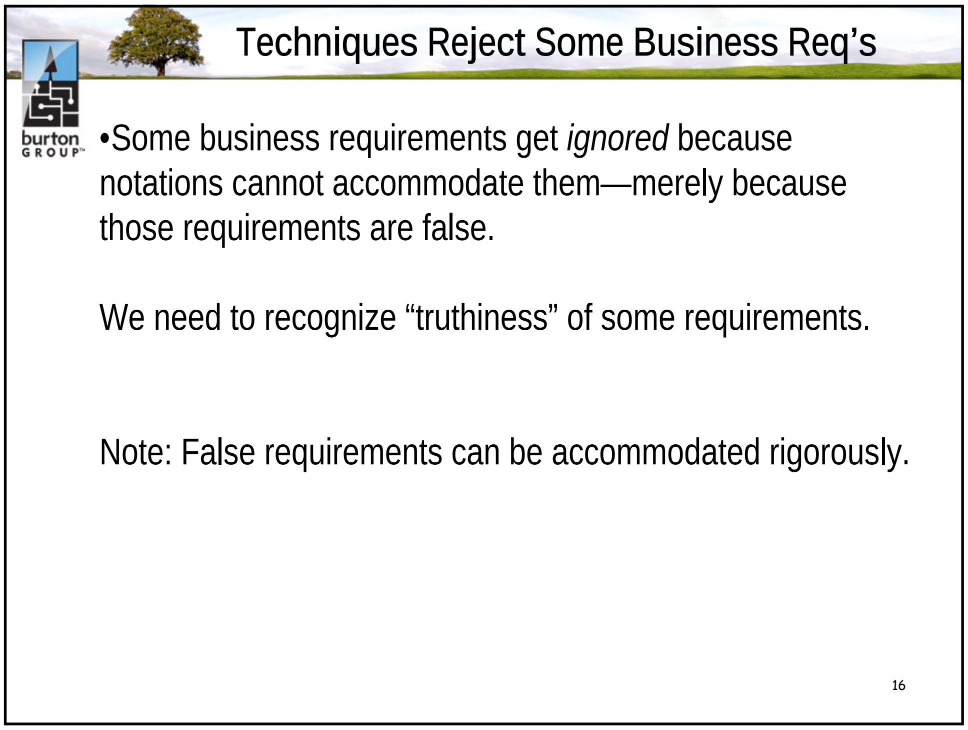
Conflict:

- Modeling Notations Omit Business Motivations
- *Modeling Notations Scorn Some Requirements Entirely*
- Software Development Suffers

Resolution:

- What You Can Do About It

15



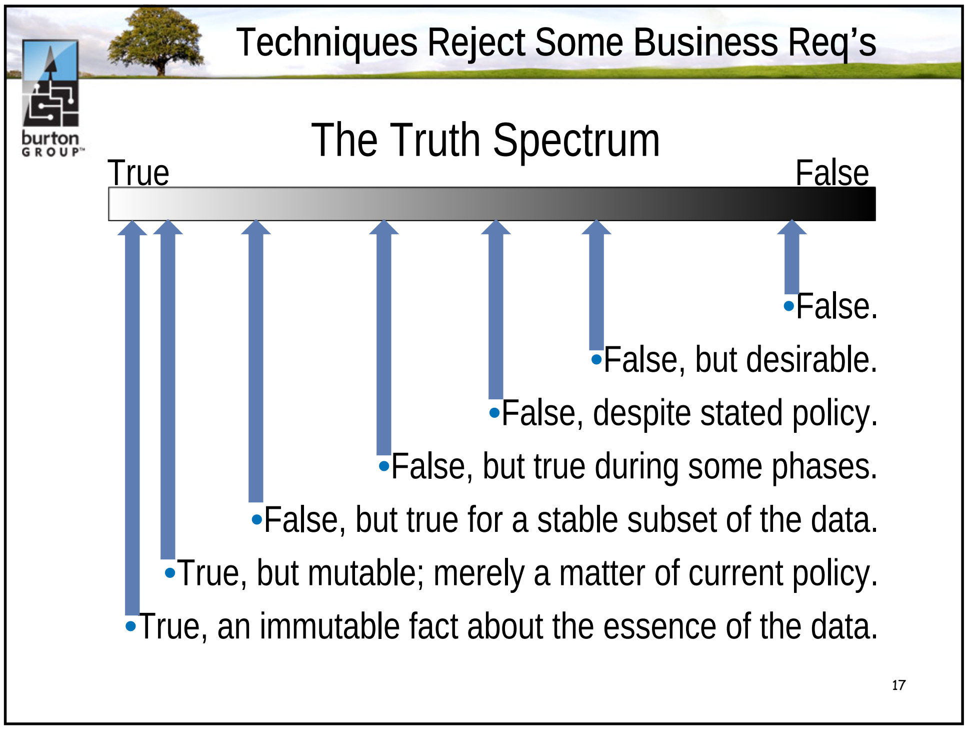
Techniques Reject Some Business Req's

•Some business requirements get *ignored* because notations cannot accommodate them—merely because those requirements are false.

We need to recognize "truthiness" of some requirements.

Note: False requirements can be accommodated rigorously.

16



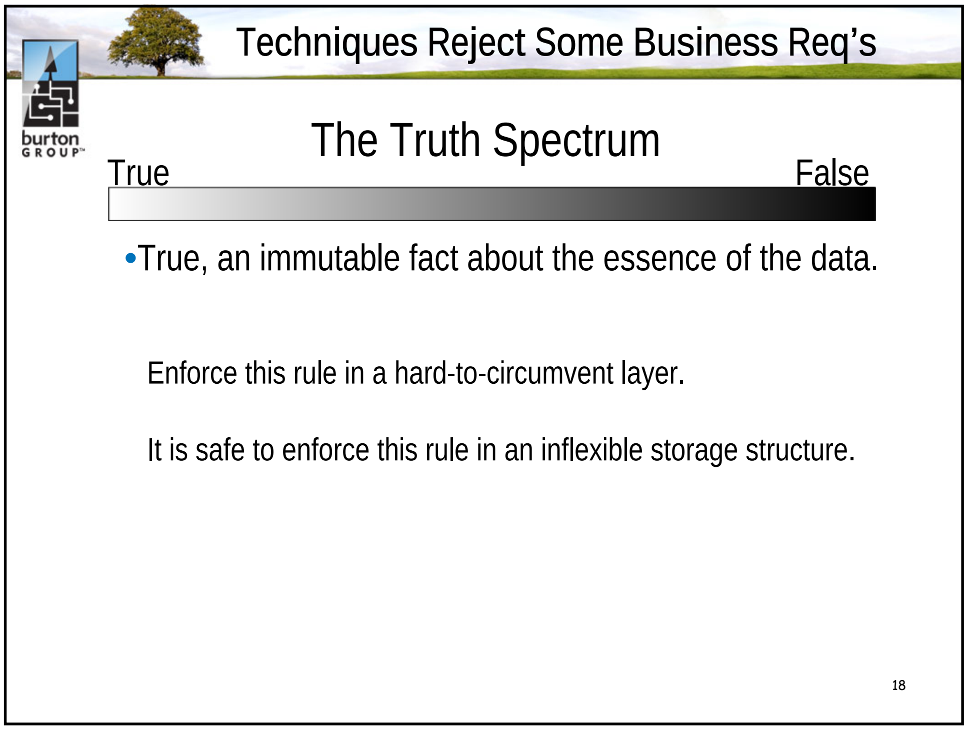
Techniques Reject Some Business Req's

The Truth Spectrum

True — False

- False.
- False, but desirable.
- False, despite stated policy.
- False, but true during some phases.
- False, but true for a stable subset of the data.
- True, but mutable; merely a matter of current policy.
- True, an immutable fact about the essence of the data.

17



Techniques Reject Some Business Req's

The Truth Spectrum

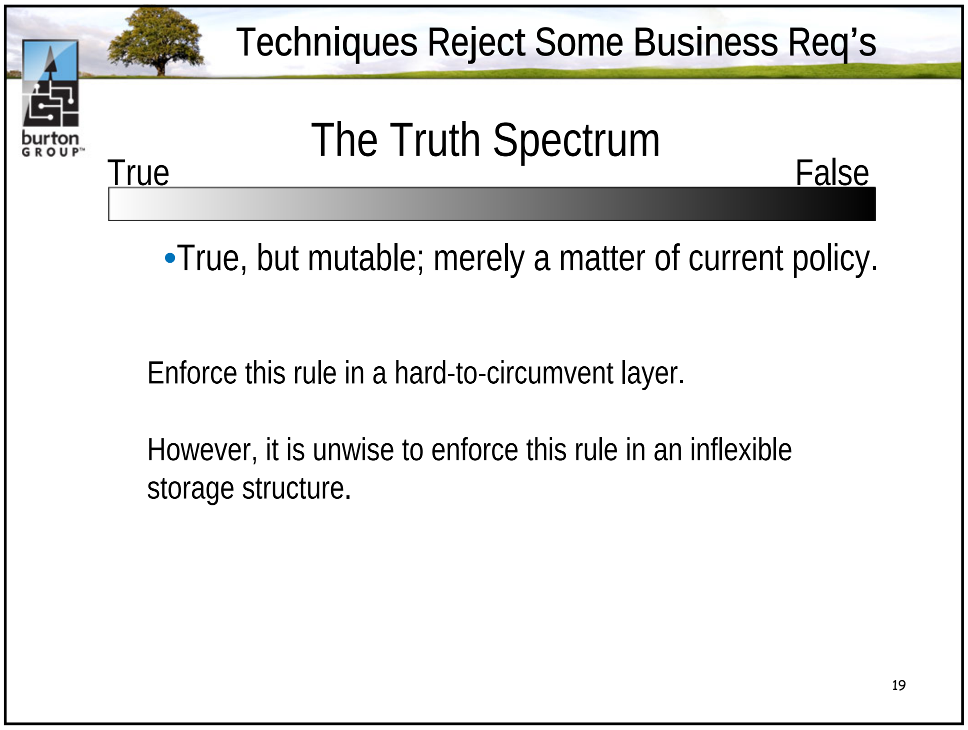
True — False

- True, an immutable fact about the essence of the data.

Enforce this rule in a hard-to-circumvent layer.

It is safe to enforce this rule in an inflexible storage structure.

18



Techniques Reject Some Business Req's

The Truth Spectrum

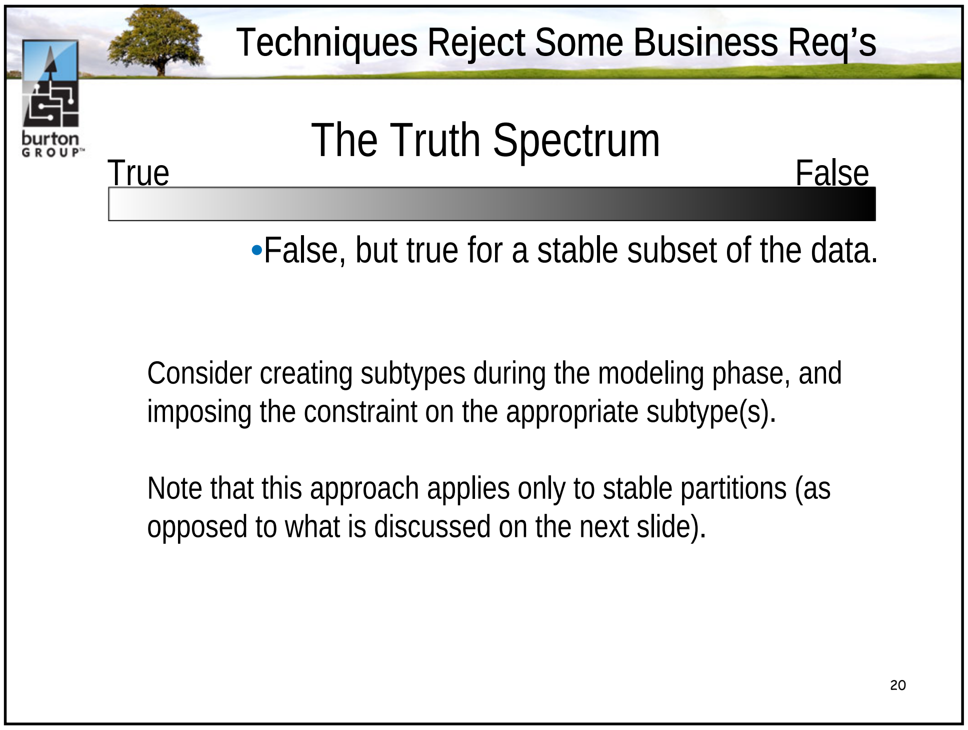
True False

•True, but mutable; merely a matter of current policy.

Enforce this rule in a hard-to-circumvent layer.

However, it is unwise to enforce this rule in an inflexible storage structure.

19



Techniques Reject Some Business Req's

The Truth Spectrum

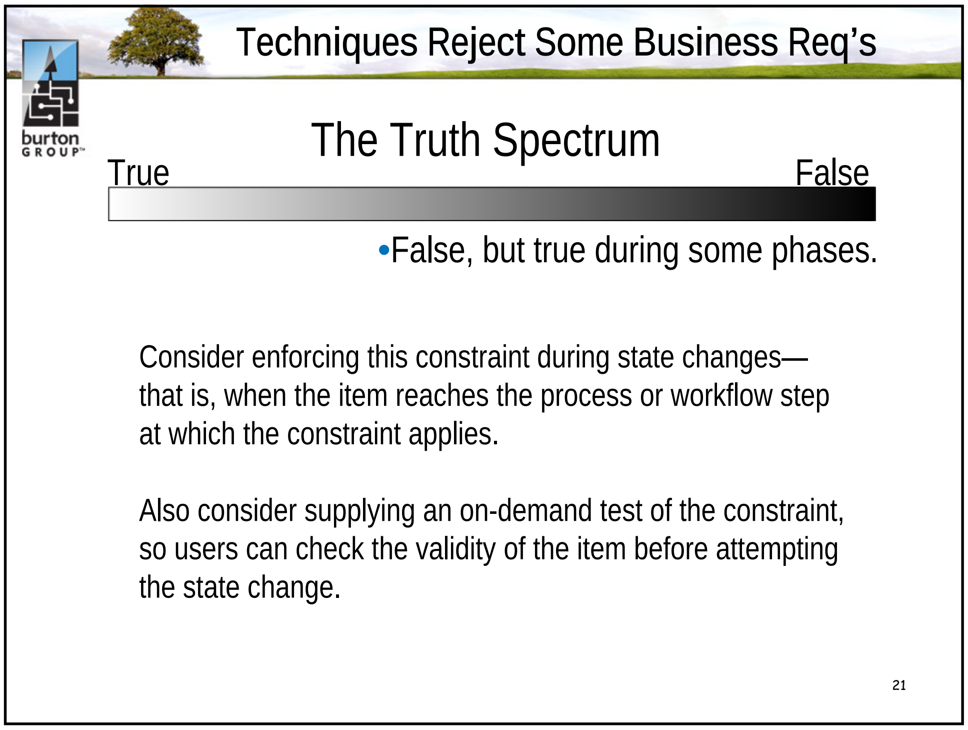
True False

•False, but true for a stable subset of the data.

Consider creating subtypes during the modeling phase, and imposing the constraint on the appropriate subtype(s).

Note that this approach applies only to stable partitions (as opposed to what is discussed on the next slide).

20



Techniques Reject Some Business Req’s

The Truth Spectrum

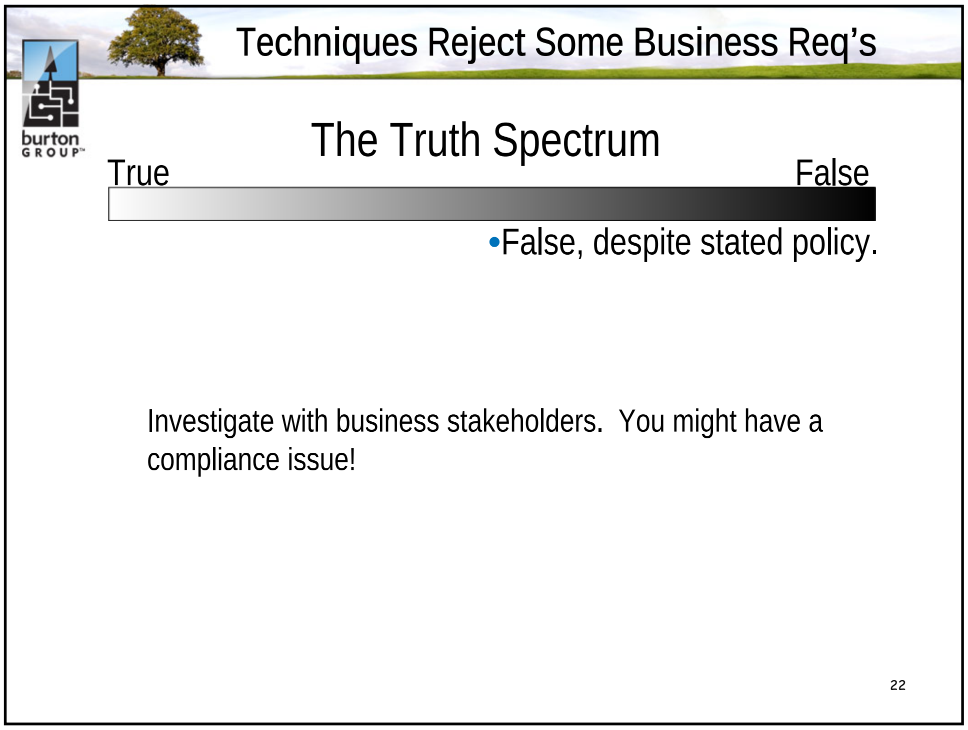
True — False

• False, but true during some phases.

Consider enforcing this constraint during state changes—that is, when the item reaches the process or workflow step at which the constraint applies.

Also consider supplying an on-demand test of the constraint, so users can check the validity of the item before attempting the state change.

21



Techniques Reject Some Business Req’s

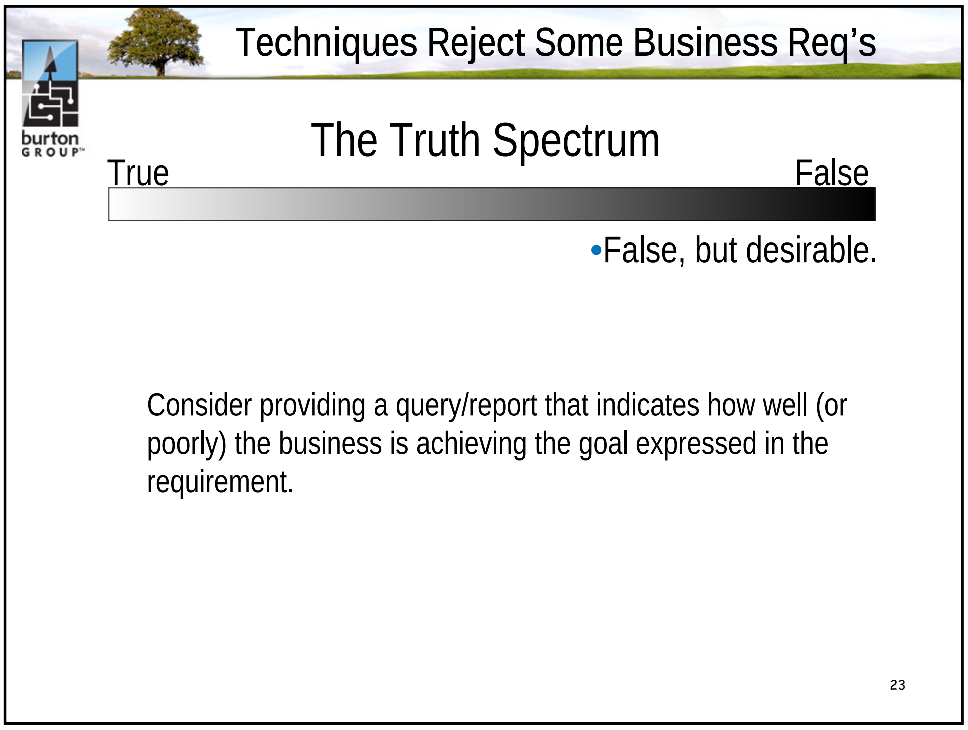
The Truth Spectrum

True — False

• False, despite stated policy.

Investigate with business stakeholders. You might have a compliance issue!

22



Techniques Reject Some Business Req's

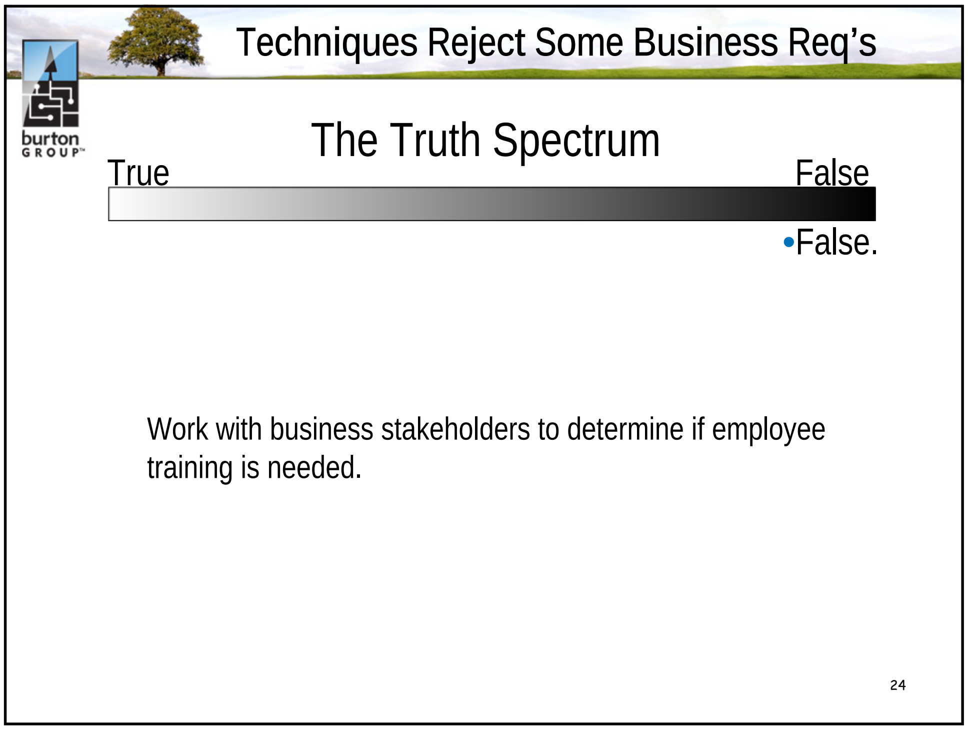
The Truth Spectrum

True — False

• False, but desirable.

Consider providing a query/report that indicates how well (or poorly) the business is achieving the goal expressed in the requirement.

23



Techniques Reject Some Business Req's

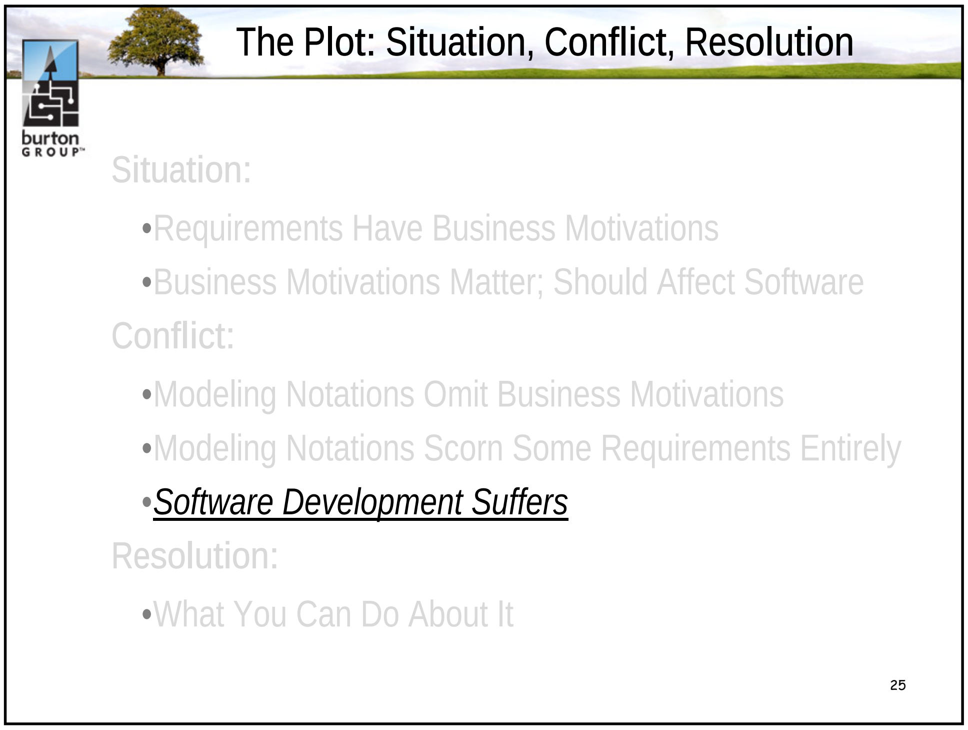
The Truth Spectrum

True — False

• False.

Work with business stakeholders to determine if employee training is needed.

24



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

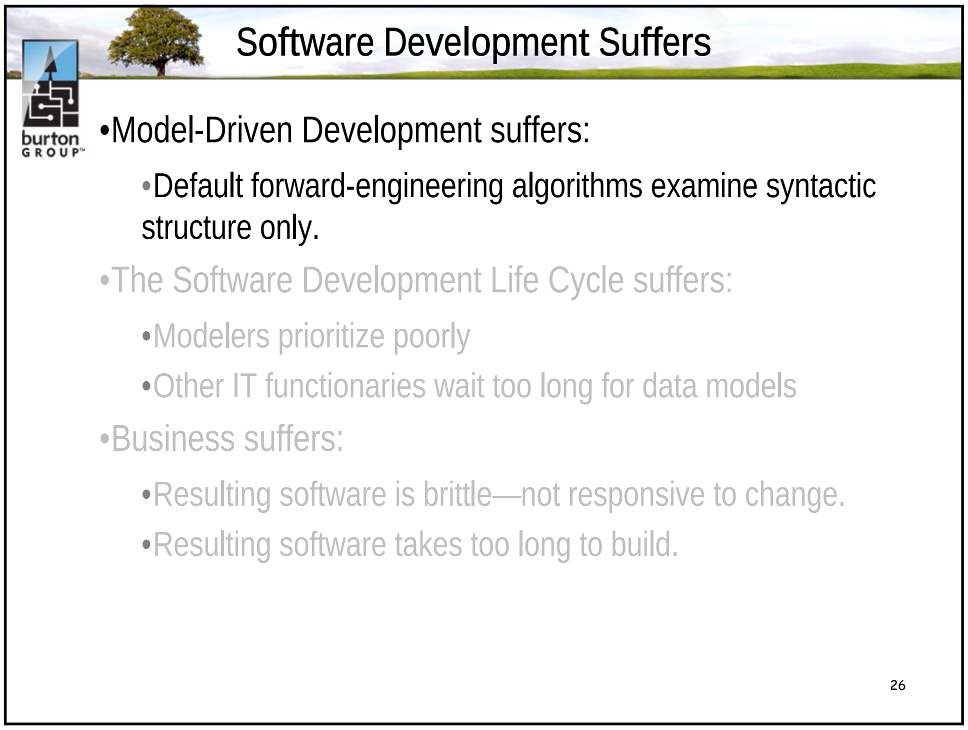
Conflict:

- Modeling Notations Omit Business Motivations
- Modeling Notations Scorn Some Requirements Entirely
- *Software Development Suffers*

Resolution:

- What You Can Do About It

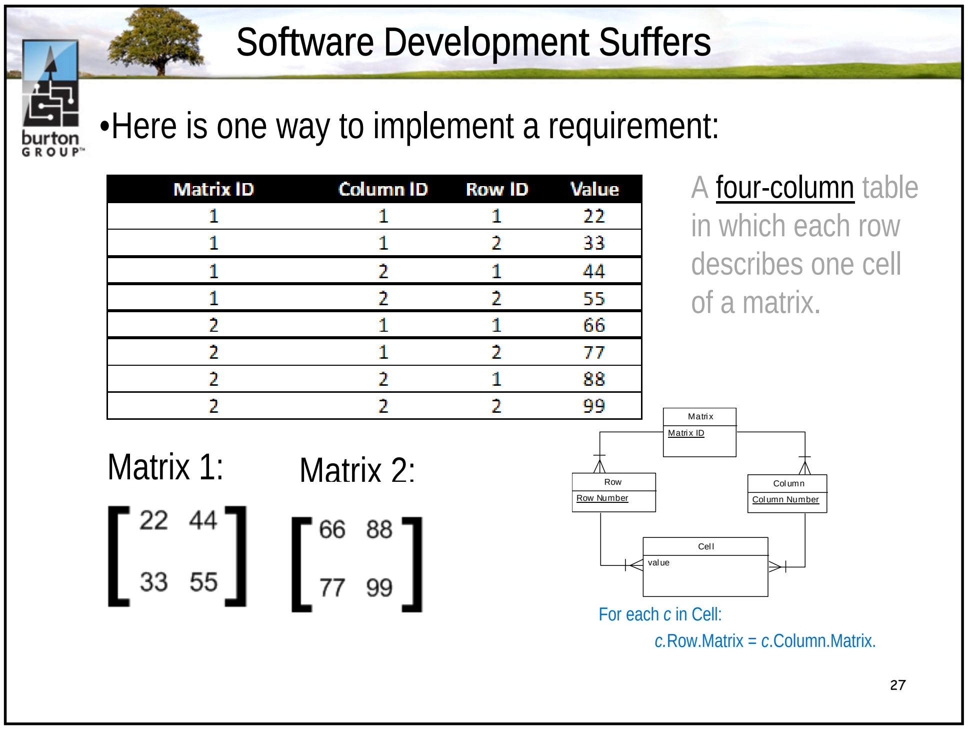
25



Software Development Suffers

- Model-Driven Development suffers:
  - Default forward-engineering algorithms examine syntactic structure only.
- The Software Development Life Cycle suffers:
  - Modelers prioritize poorly
  - Other IT functionaries wait too long for data models
- Business suffers:
  - Resulting software is brittle—not responsive to change.
  - Resulting software takes too long to build.

26



Software Development Suffers

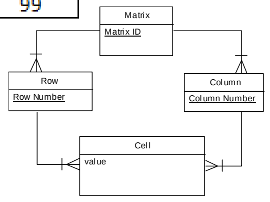
•Here is one way to implement a requirement:

Matrix ID	Column ID	Row ID	Value
1	1	1	22
1	1	2	33
1	2	1	44
1	2	2	55
2	1	1	66
2	1	2	77
2	2	1	88
2	2	2	99

A four-column table in which each row describes one cell of a matrix.

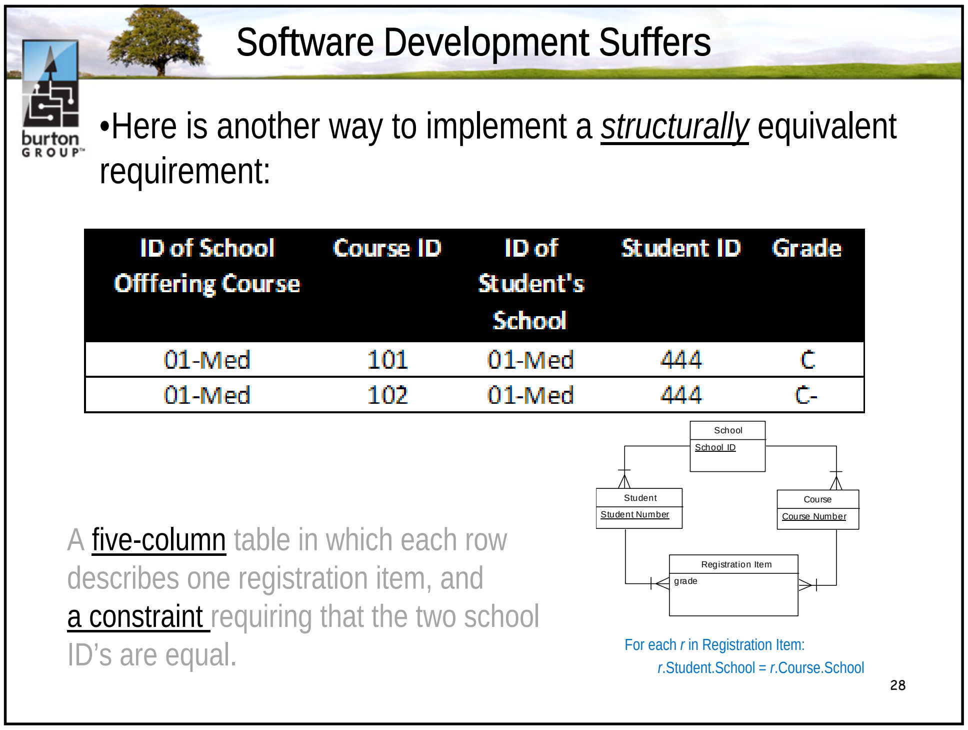
Matrix 1: $\begin{bmatrix} 22 & 44 \\ 33 & 55 \end{bmatrix}$

Matrix 2: $\begin{bmatrix} 66 & 88 \\ 77 & 99 \end{bmatrix}$



For each *c* in Cell:
c.Row.Matrix = *c*.Column.Matrix.

27

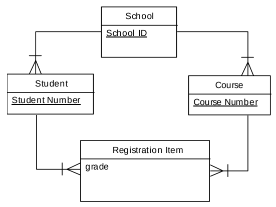


Software Development Suffers

•Here is another way to implement a *structurally* equivalent requirement:

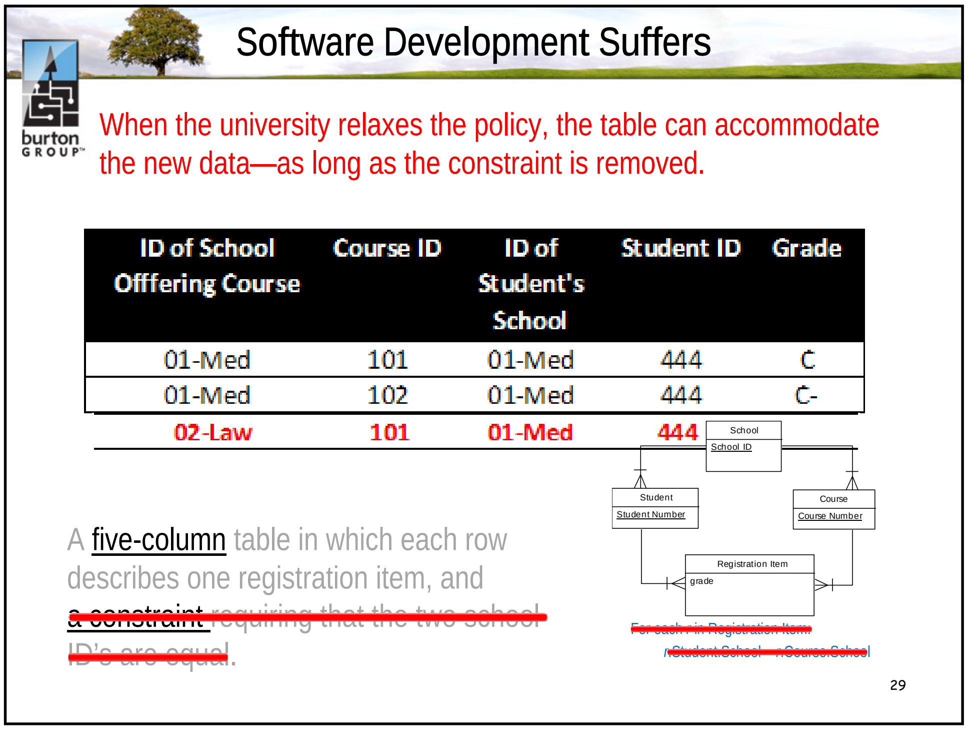
ID of School Offering Course	Course ID	ID of Student's School	Student ID	Grade
01-Med	101	01-Med	444	C
01-Med	102	01-Med	444	C-

A five-column table in which each row describes one registration item, and a constraint requiring that the two school ID's are equal.



For each *r* in Registration Item:
r.Student.School = *r*.Course.School

28

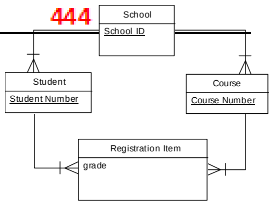


Software Development Suffers

When the university relaxes the policy, the table can accommodate the new data—as long as the constraint is removed.

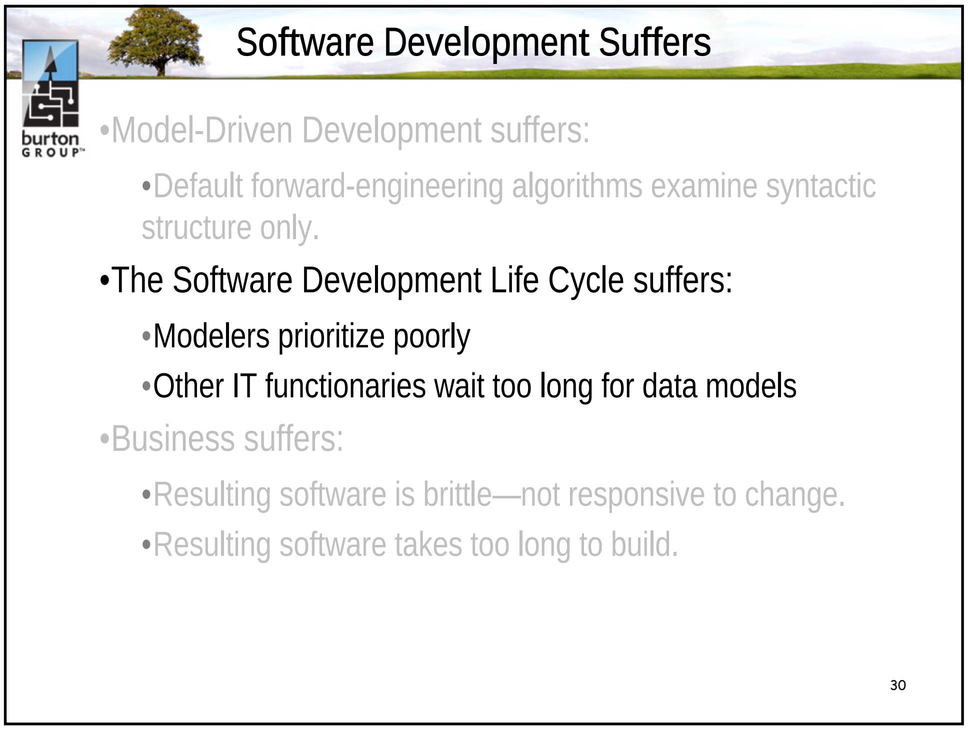
ID of School Offering Course	Course ID	ID of Student's School	Student ID	Grade
01-Med	101	01-Med	444	C
01-Med	102	01-Med	444	C-
02-Law	101	01-Med	444	

A five-column table in which each row describes one registration item, and ~~a constraint requiring that the two school ID's are equal~~.



~~For each r in Registration Item:~~
~~r.Student.School = r.Course.School~~

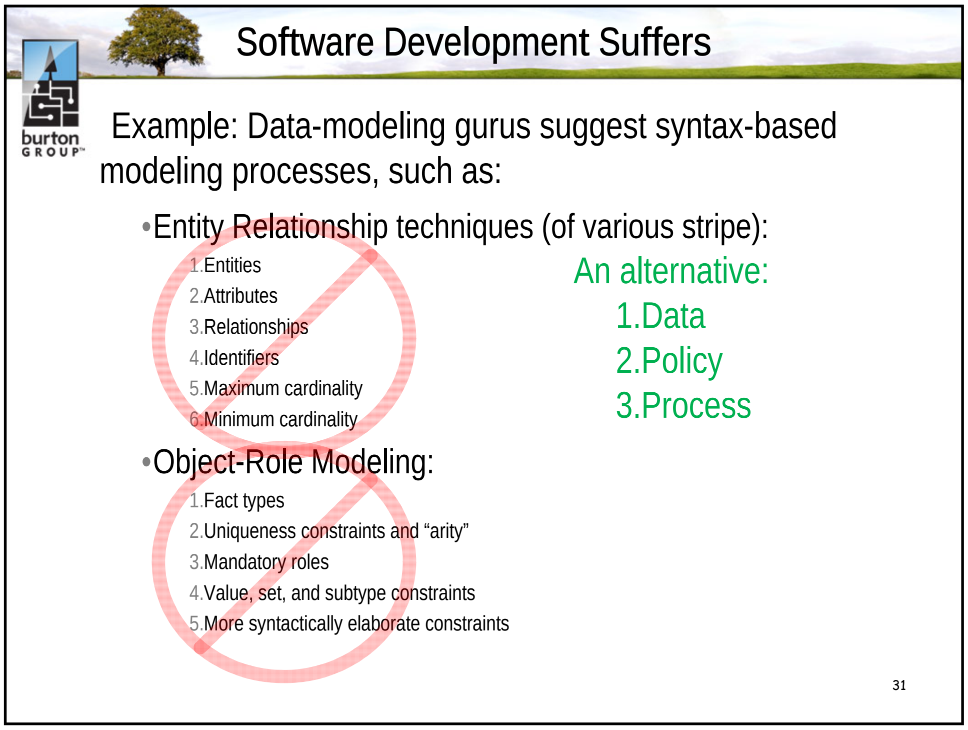
29



Software Development Suffers

- Model-Driven Development suffers:
  - Default forward-engineering algorithms examine syntactic structure only.
- The Software Development Life Cycle suffers:
  - Modelers prioritize poorly
  - Other IT functionaries wait too long for data models
- Business suffers:
  - Resulting software is brittle—not responsive to change.
  - Resulting software takes too long to build.

30



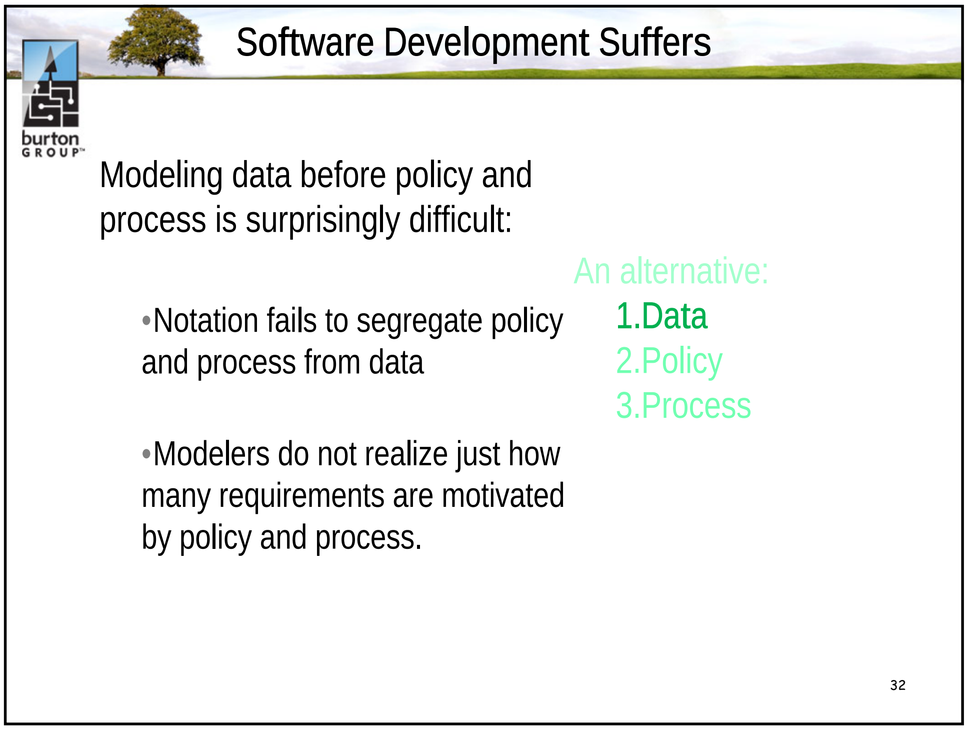
Software Development Suffers

Example: Data-modeling gurus suggest syntax-based modeling processes, such as:

- Entity Relationship techniques (of various stripe):
  1. Entities
  2. Attributes
  3. Relationships
  4. Identifiers
  5. Maximum cardinality
  6. Minimum cardinality
- Object-Role Modeling:
  1. Fact types
  2. Uniqueness constraints and “arity”
  3. Mandatory roles
  4. Value, set, and subtype constraints
  5. More syntactically elaborate constraints

An alternative:
1. Data
2. Policy
3. Process

31



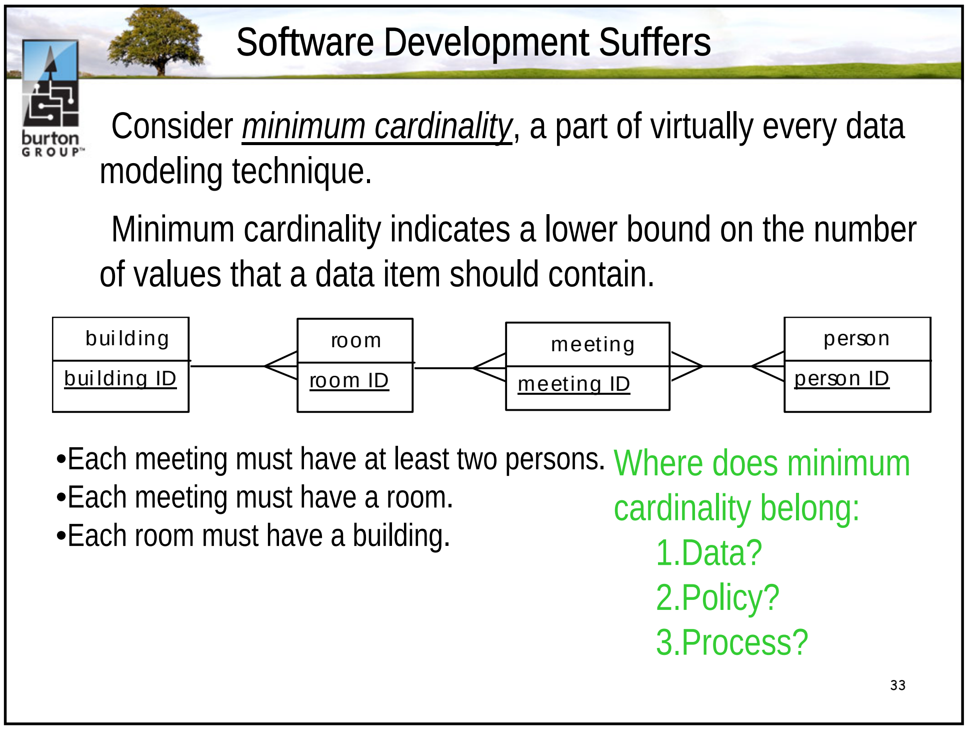
Software Development Suffers

Modeling data before policy and process is surprisingly difficult:

- Notation fails to segregate policy and process from data
- Modelers do not realize just how many requirements are motivated by policy and process.

An alternative:
1. Data
2. Policy
3. Process

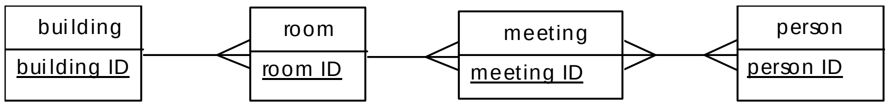
32



Software Development Suffers

Consider *minimum cardinality*, a part of virtually every data modeling technique.

Minimum cardinality indicates a lower bound on the number of values that a data item should contain.

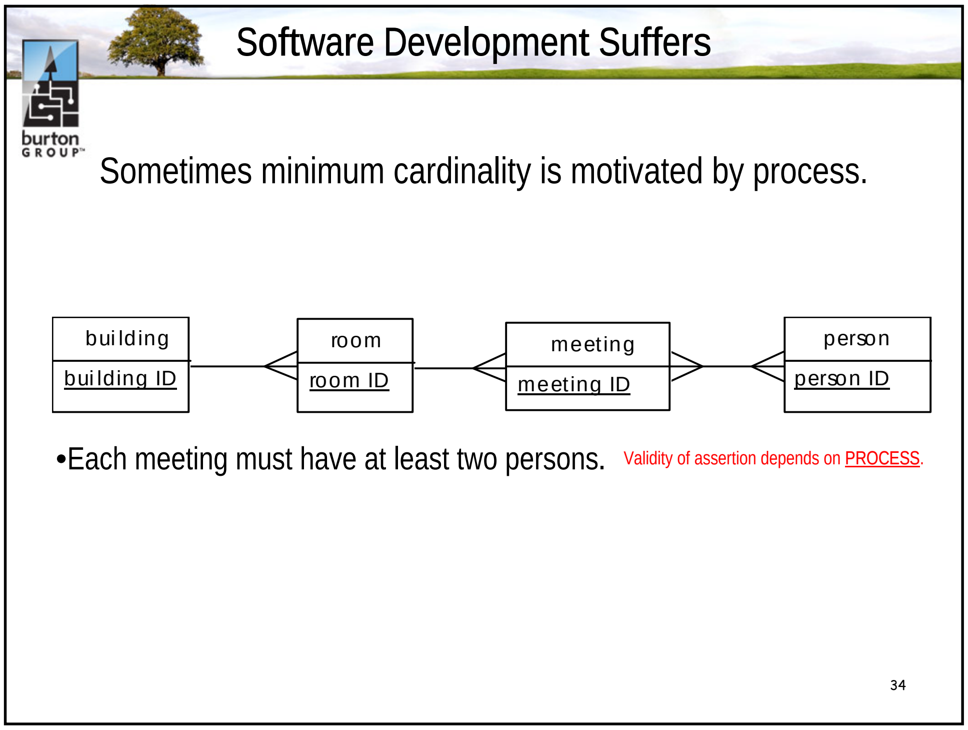


•Each meeting must have at least two persons.
•Each meeting must have a room.
•Each room must have a building.

Where does minimum cardinality belong:

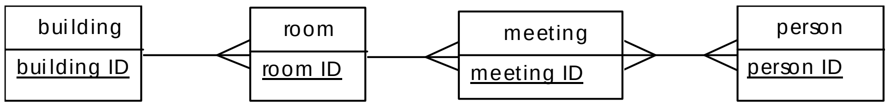
1. Data?
2. Policy?
3. Process?

33



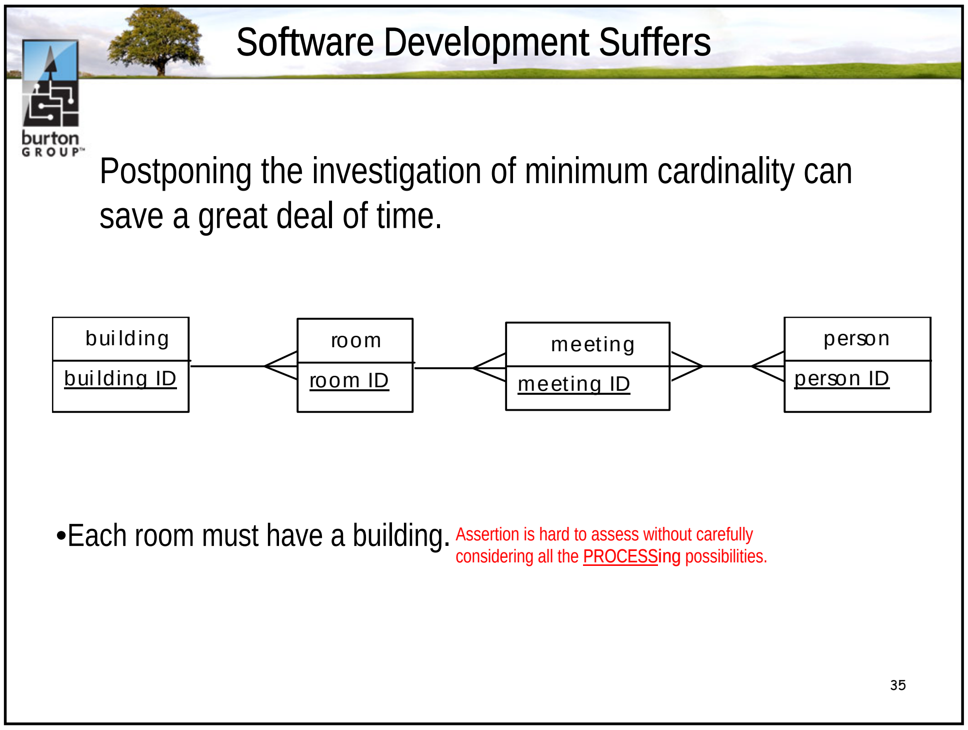
Software Development Suffers

Sometimes minimum cardinality is motivated by process.



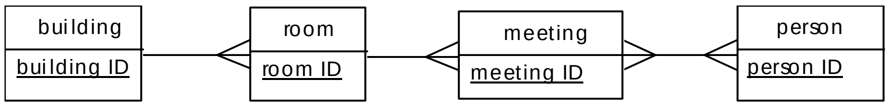
•Each meeting must have at least two persons. Validity of assertion depends on PROCESS.

34



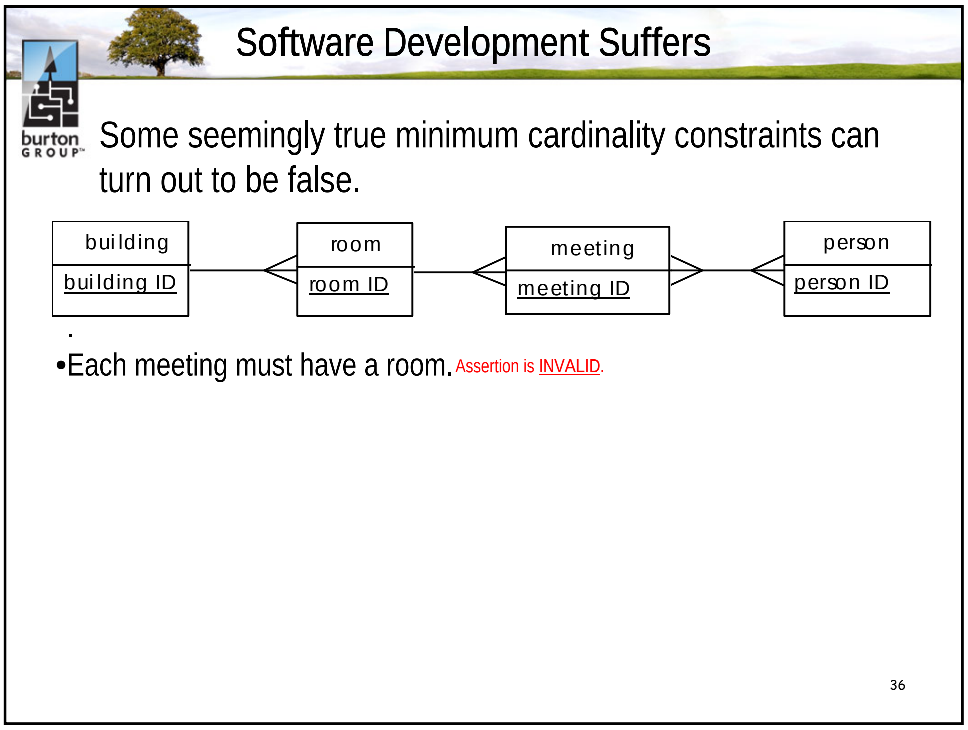
Software Development Suffers

Postponing the investigation of minimum cardinality can save a great deal of time.



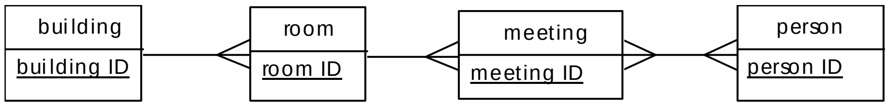
•Each room must have a building. Assertion is hard to assess without carefully considering all the PROCESSing possibilities.

35



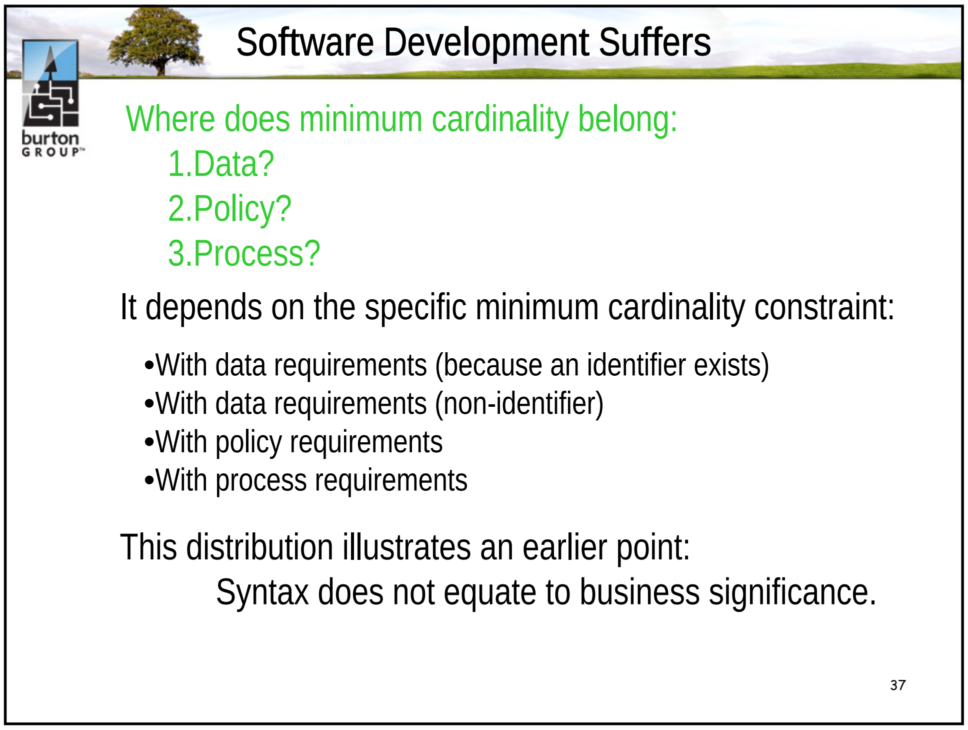
Software Development Suffers

Some seemingly true minimum cardinality constraints can turn out to be false.



•Each meeting must have a room. Assertion is INVALID.

36



Software Development Suffers

Where does minimum cardinality belong:

1. Data?
2. Policy?
3. Process?

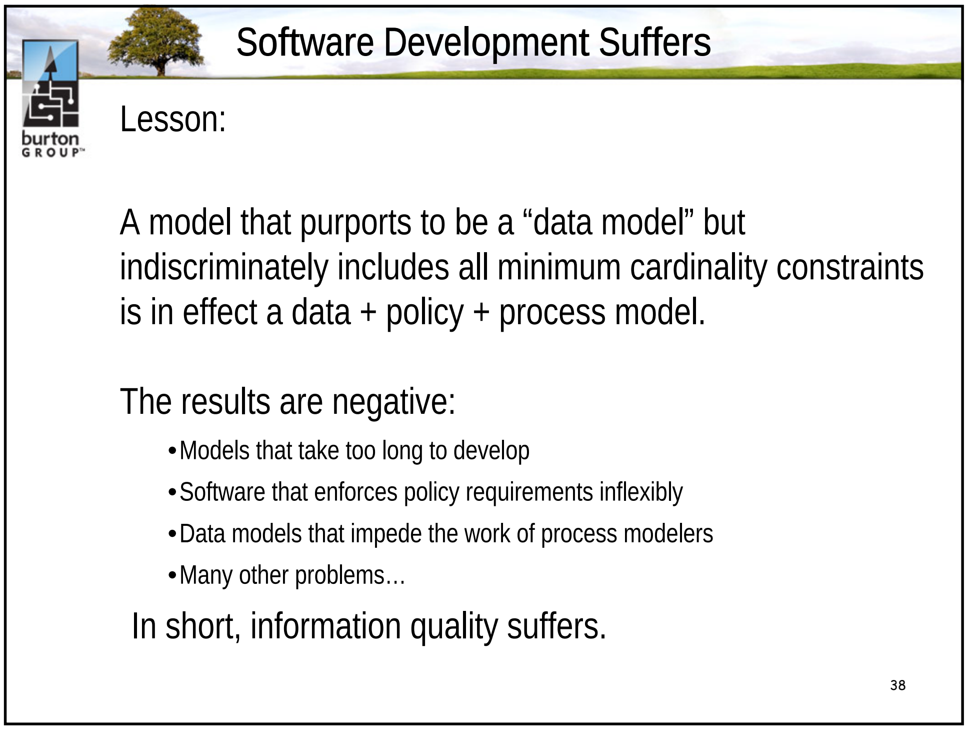
It depends on the specific minimum cardinality constraint:

- With data requirements (because an identifier exists)
- With data requirements (non-identifier)
- With policy requirements
- With process requirements

This distribution illustrates an earlier point:

Syntax does not equate to business significance.

37



Software Development Suffers

Lesson:

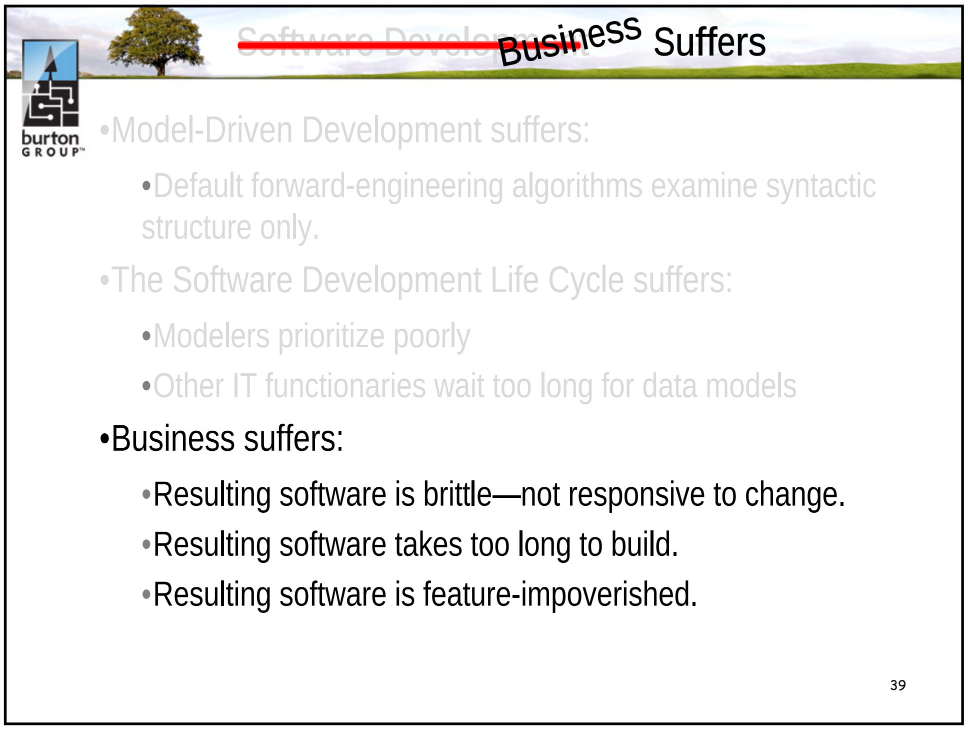
A model that purports to be a “data model” but indiscriminately includes all minimum cardinality constraints is in effect a data + policy + process model.

The results are negative:

- Models that take too long to develop
- Software that enforces policy requirements inflexibly
- Data models that impede the work of process modelers
- Many other problems…

In short, information quality suffers.

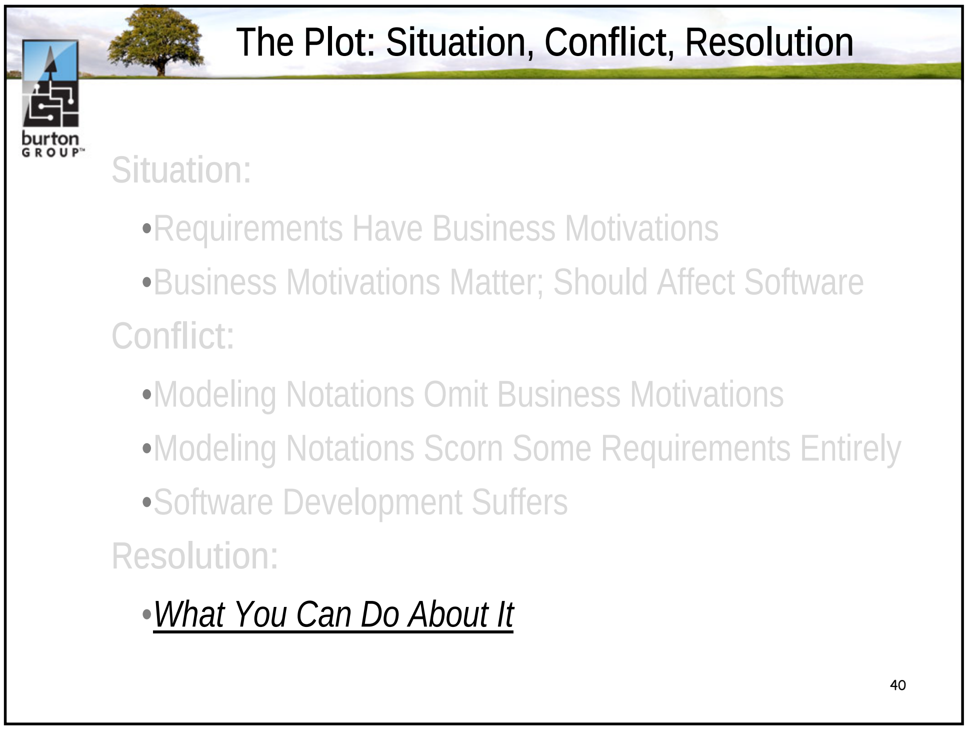
38



~~Software Development~~ Business Suffers

- Model-Driven Development suffers:
  - Default forward-engineering algorithms examine syntactic structure only.
- The Software Development Life Cycle suffers:
  - Modelers prioritize poorly
  - Other IT functionaries wait too long for data models
- Business suffers:
  - Resulting software is brittle—not responsive to change.
  - Resulting software takes too long to build.
  - Resulting software is feature-impoverished.

39



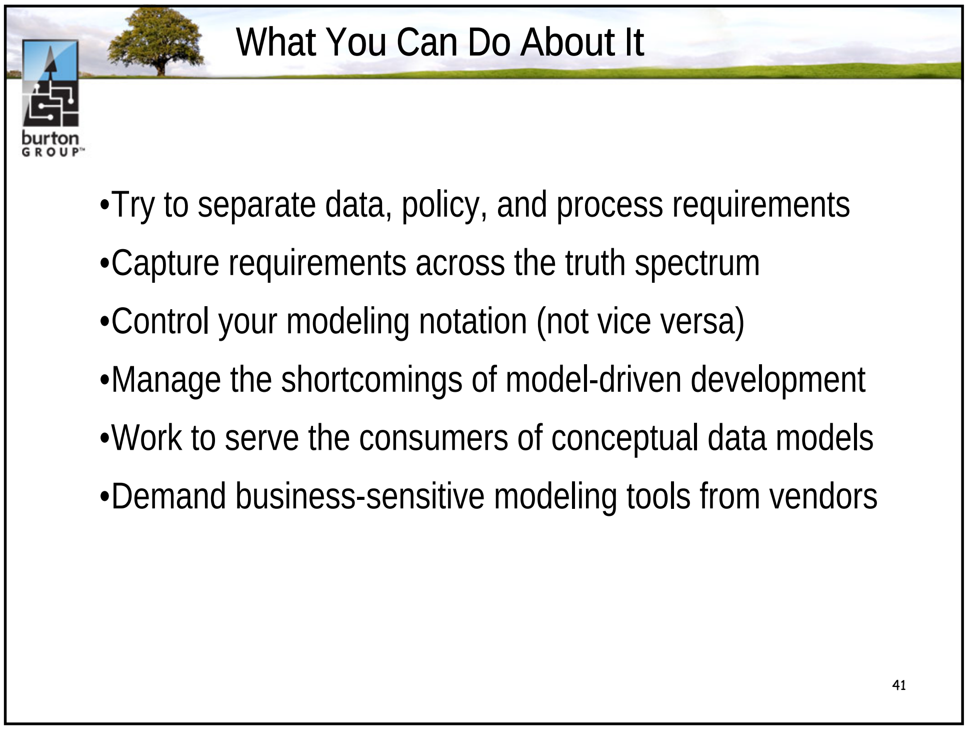
The Plot: Situation, Conflict, Resolution

Situation:
- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

Conflict:
- Modeling Notations Omit Business Motivations
- Modeling Notations Scorn Some Requirements Entirely
- Software Development Suffers

Resolution:
- *What You Can Do About It*

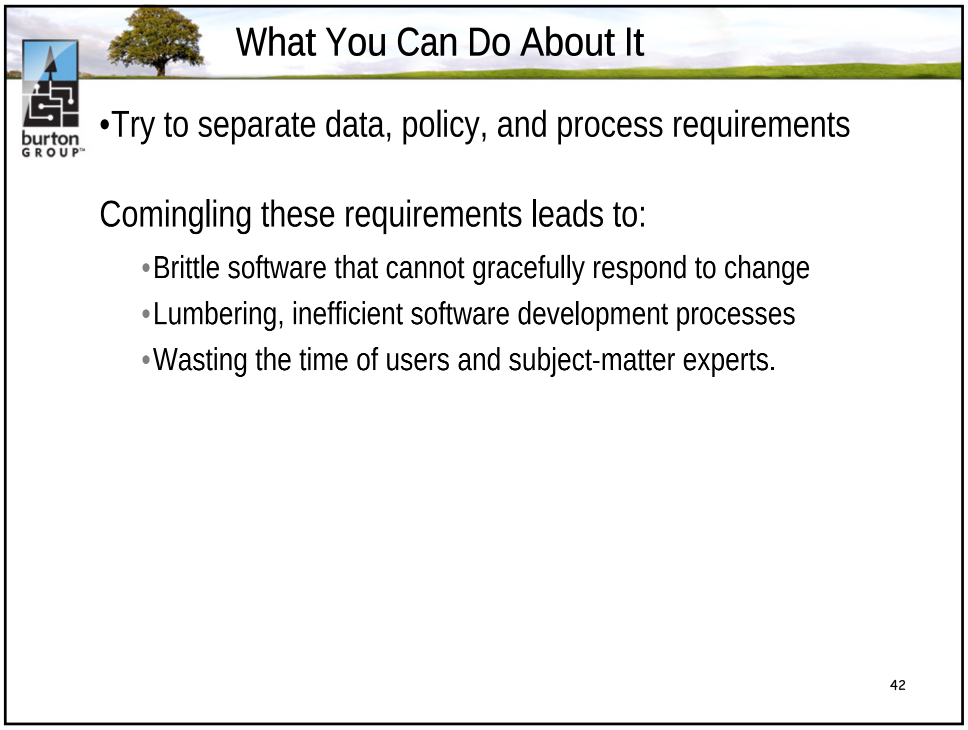
40



What You Can Do About It

- Try to separate data, policy, and process requirements
- Capture requirements across the truth spectrum
- Control your modeling notation (not vice versa)
- Manage the shortcomings of model-driven development
- Work to serve the consumers of conceptual data models
- Demand business-sensitive modeling tools from vendors

41



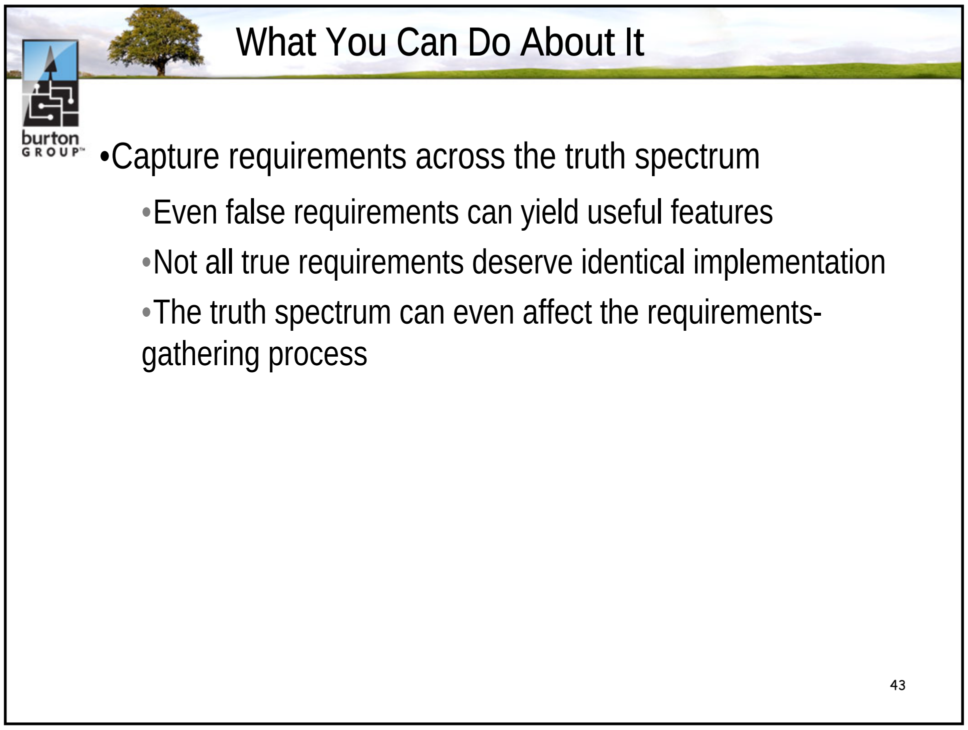
What You Can Do About It

- Try to separate data, policy, and process requirements

Comingling these requirements leads to:

- Brittle software that cannot gracefully respond to change
- Lumbering, inefficient software development processes
- Wasting the time of users and subject-matter experts.

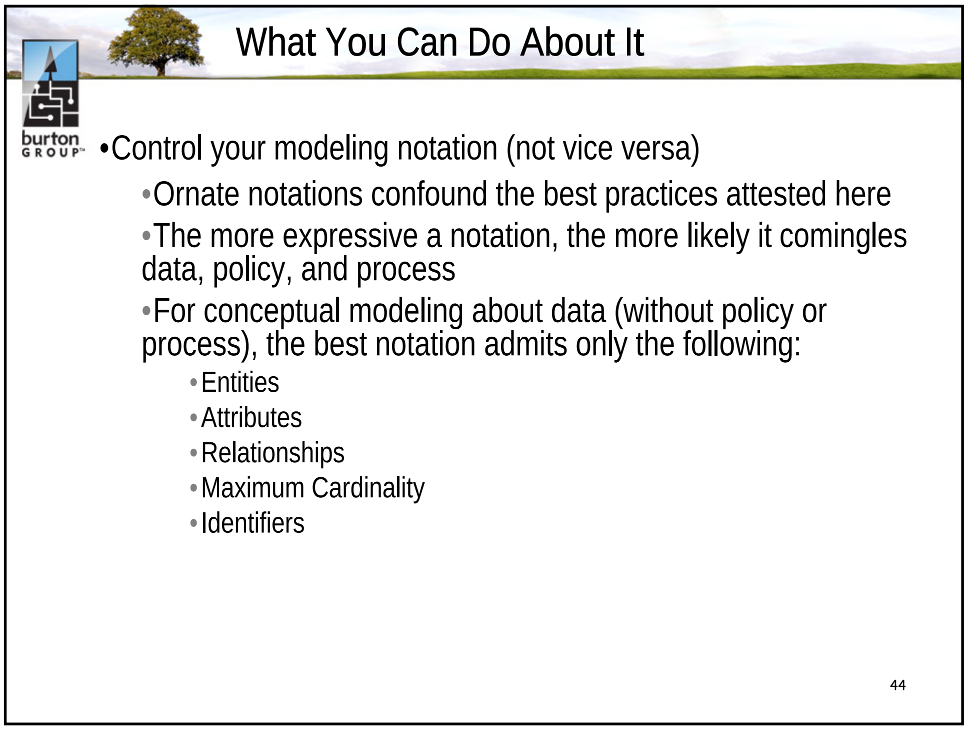
42



What You Can Do About It

- Capture requirements across the truth spectrum
  - Even false requirements can yield useful features
  - Not all true requirements deserve identical implementation
  - The truth spectrum can even affect the requirements-gathering process

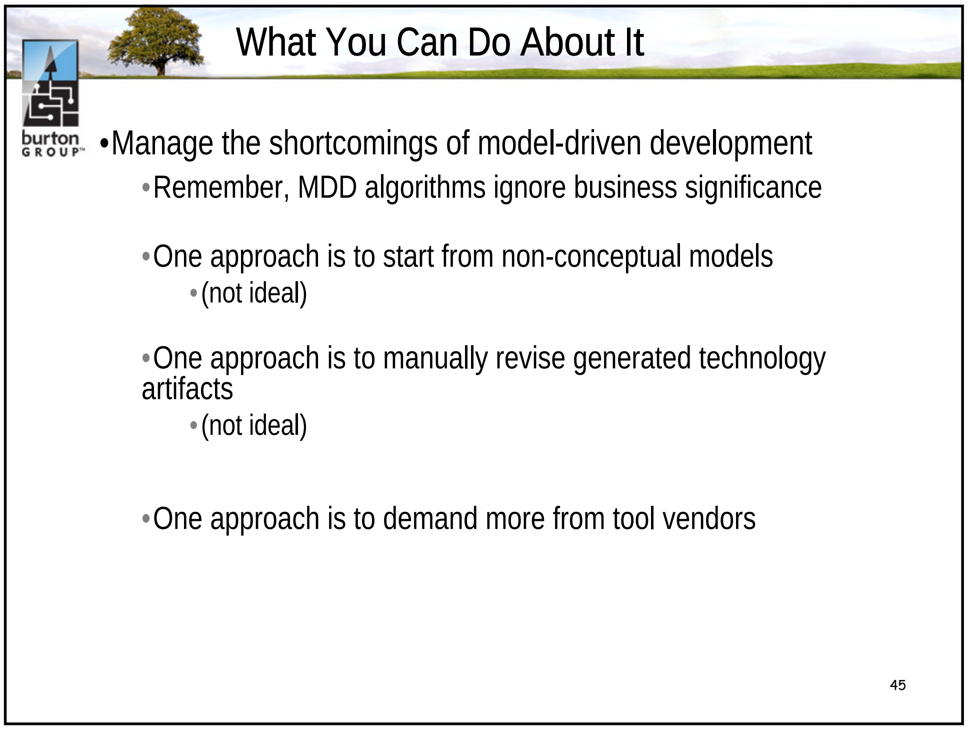
43



What You Can Do About It

- Control your modeling notation (not vice versa)
  - Ornate notations confound the best practices attested here
  - The more expressive a notation, the more likely it comingles data, policy, and process
  - For conceptual modeling about data (without policy or process), the best notation admits only the following:
    - Entities
    - Attributes
    - Relationships
    - Maximum Cardinality
    - Identifiers

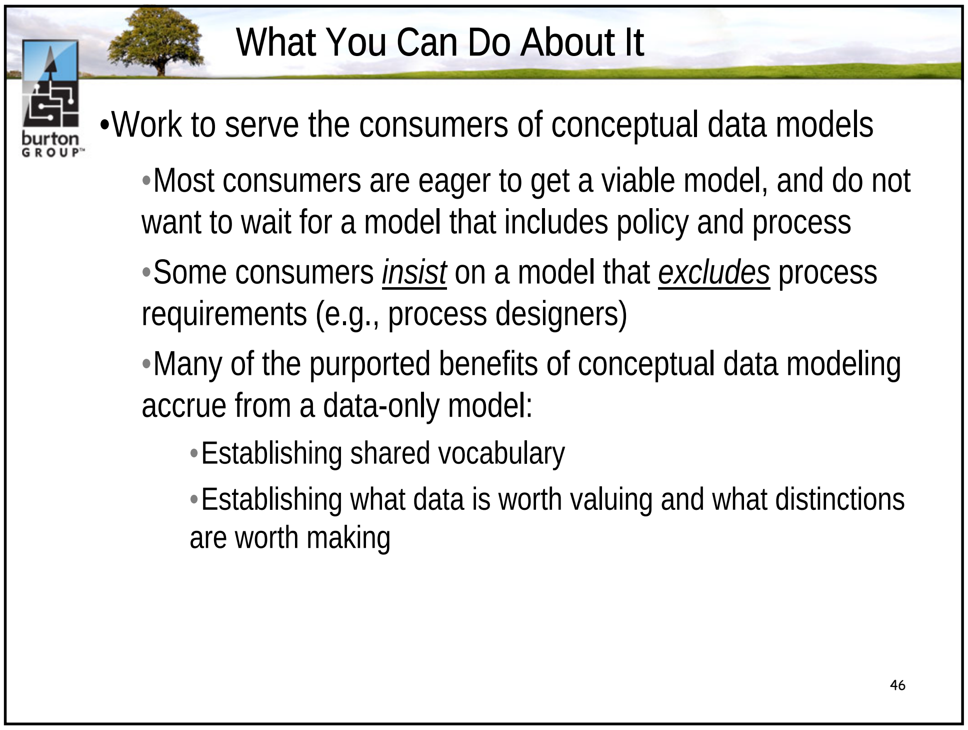
44



What You Can Do About It

- Manage the shortcomings of model-driven development
  - Remember, MDD algorithms ignore business significance
  - One approach is to start from non-conceptual models
    - (not ideal)
  - One approach is to manually revise generated technology artifacts
    - (not ideal)
  - One approach is to demand more from tool vendors

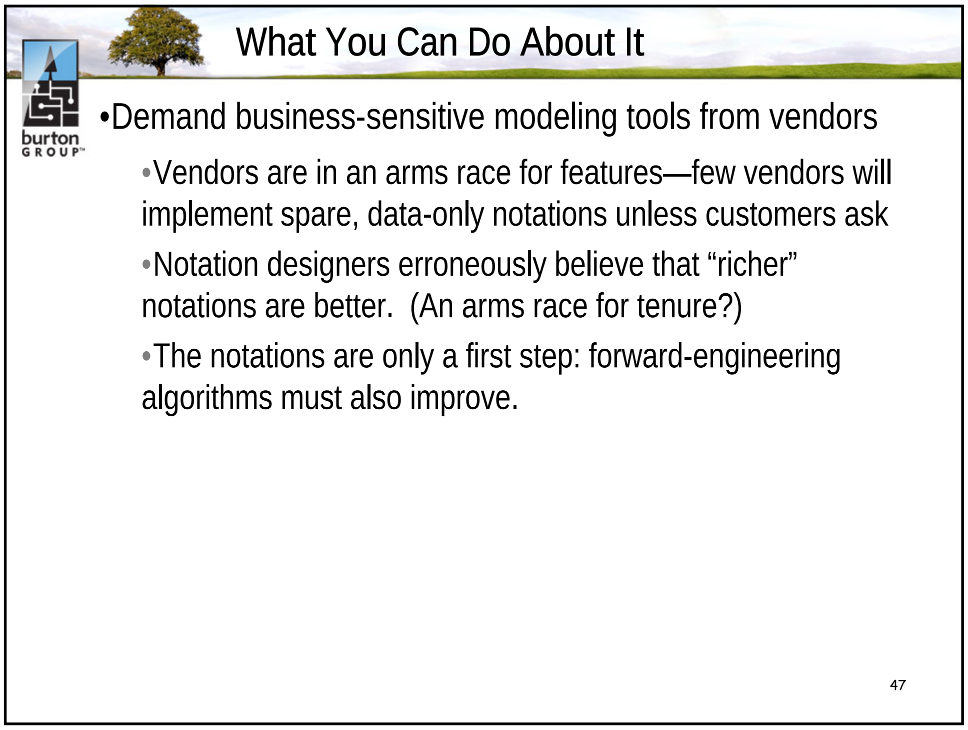
45



What You Can Do About It

- Work to serve the consumers of conceptual data models
  - Most consumers are eager to get a viable model, and do not want to wait for a model that includes policy and process
  - Some consumers *insist* on a model that *excludes* process requirements (e.g., process designers)
  - Many of the purported benefits of conceptual data modeling accrue from a data-only model:
    - Establishing shared vocabulary
    - Establishing what data is worth valuing and what distinctions are worth making

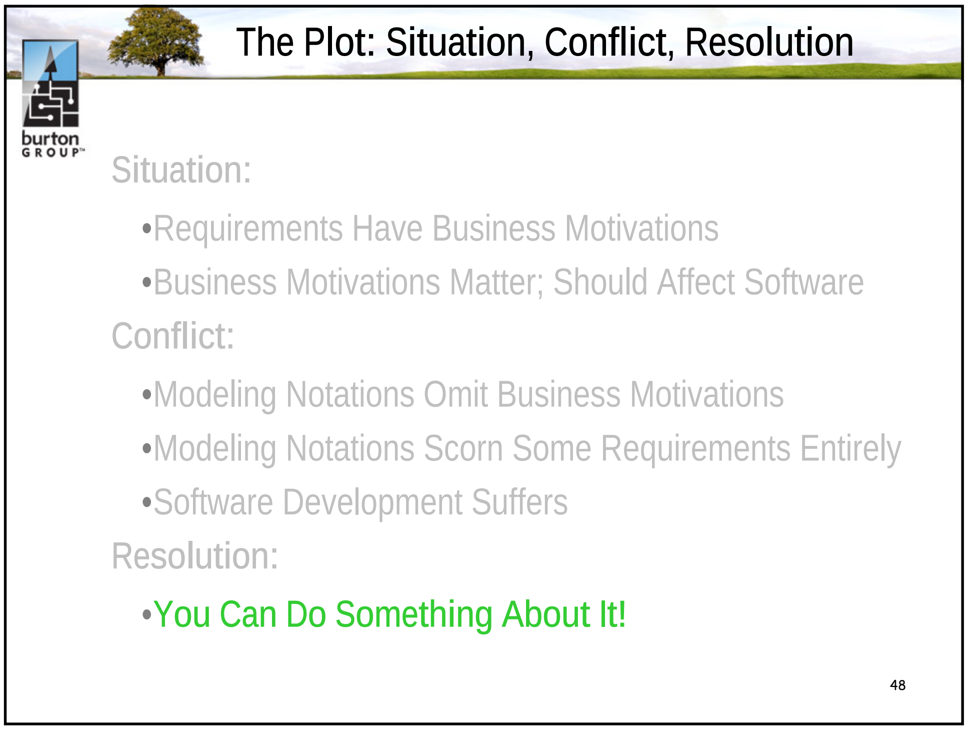
46



What You Can Do About It

- Demand business-sensitive modeling tools from vendors
  - Vendors are in an arms race for features—few vendors will implement spare, data-only notations unless customers ask
  - Notation designers erroneously believe that “richer” notations are better. (An arms race for tenure?)
  - The notations are only a first step: forward-engineering algorithms must also improve.

47



The Plot: Situation, Conflict, Resolution

Situation:

- Requirements Have Business Motivations
- Business Motivations Matter; Should Affect Software

Conflict:

- Modeling Notations Omit Business Motivations
- Modeling Notations Scorn Some Requirements Entirely
- Software Development Suffers

Resolution:

- You Can Do Something About It!

48