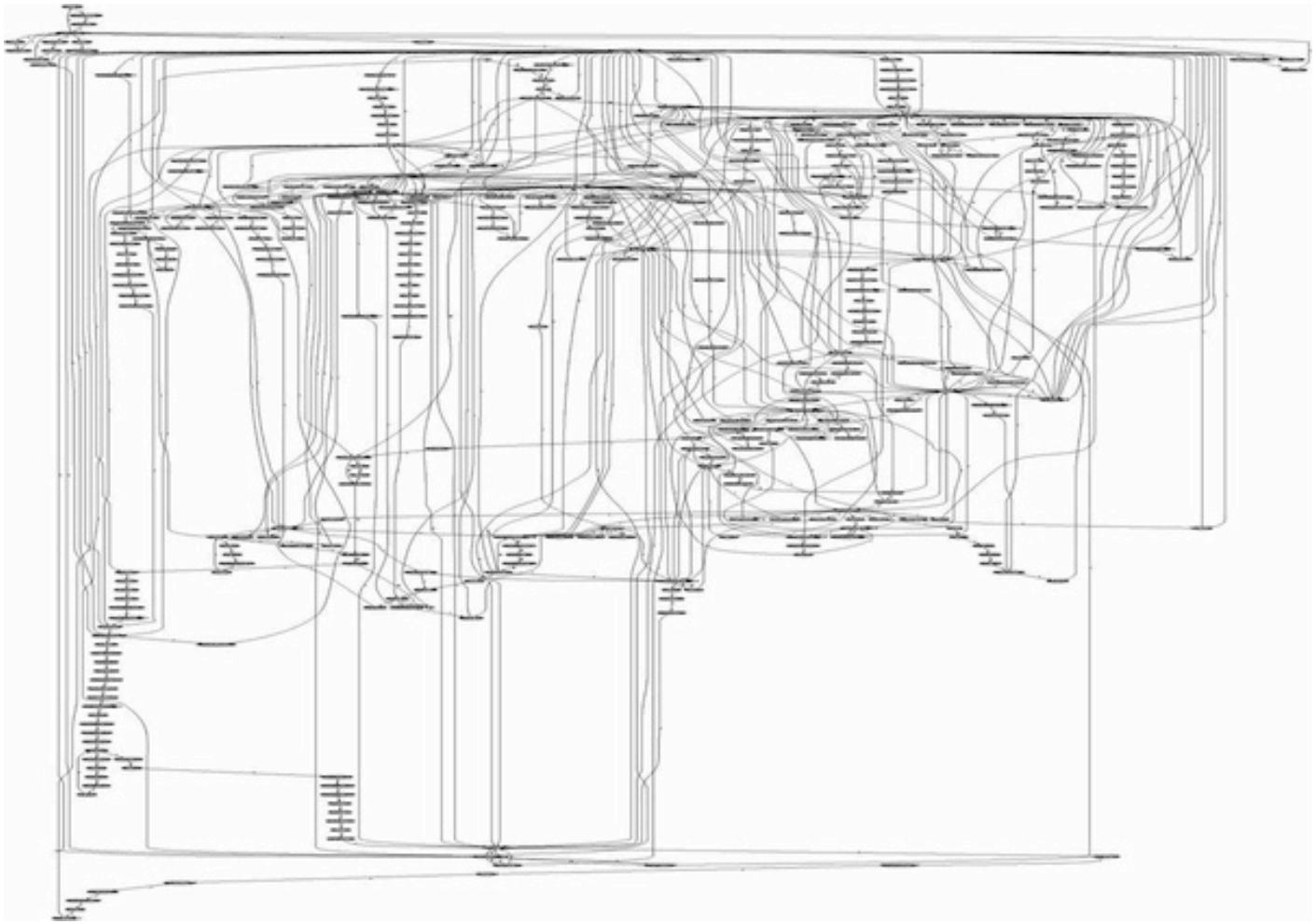
Announcements

- Quiz #1 Review and Scores
- Domain Role Viewpoint Hierarchy – Importance of separating users and customers
- Project Milestone #1 due this Friday, Oct 4th.

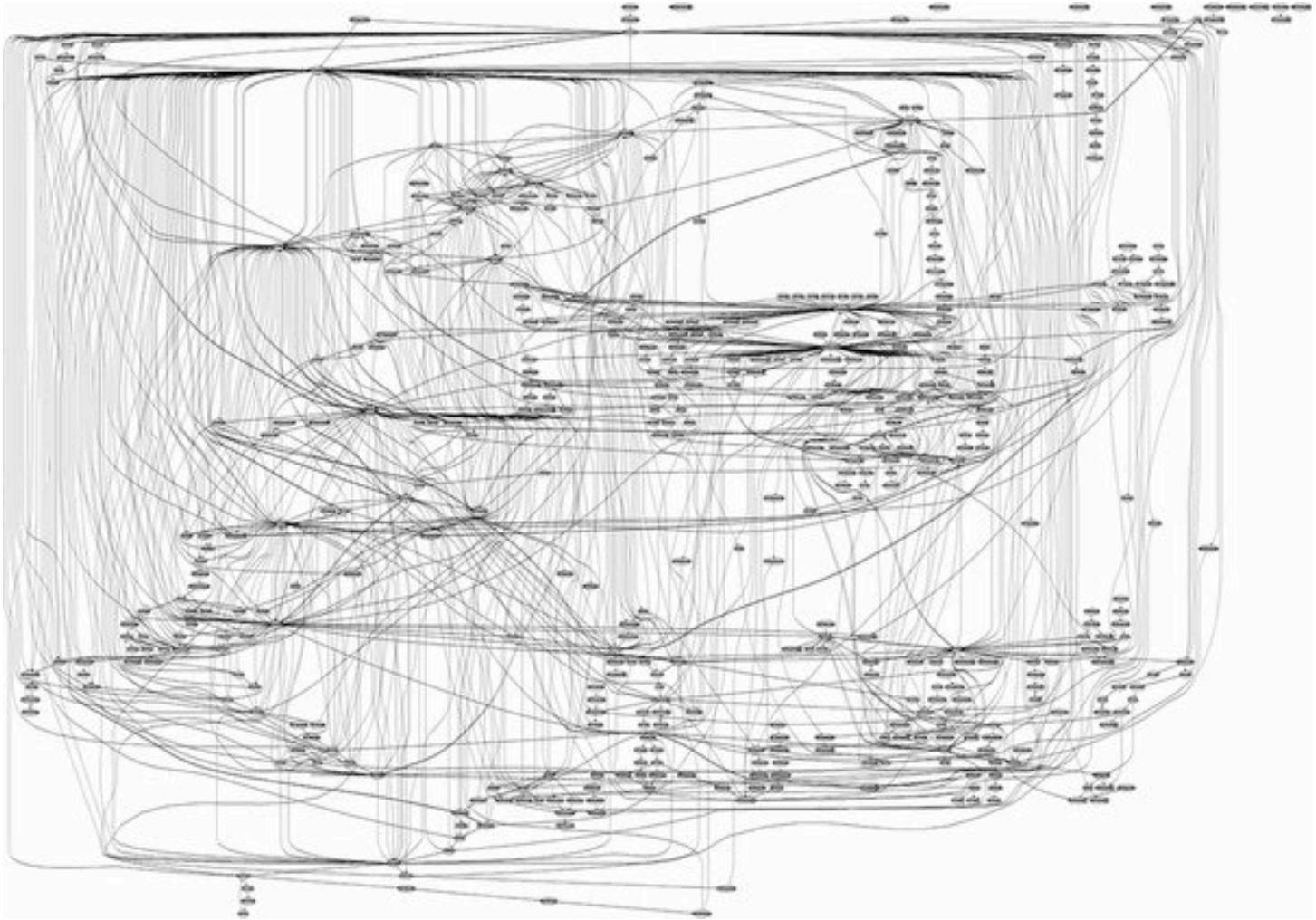
AWARENESS 3D ARCHITECTURES™: BUSINESS BLUEPRINT SPECIFICATIONS

Suzanne Barber

This is a call trace for Linux.

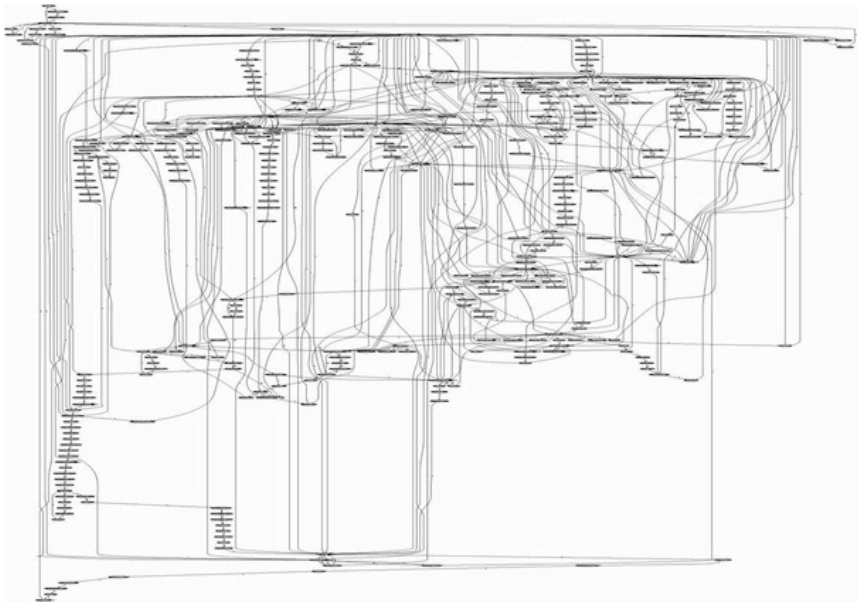


This is a call trace for Microsoft IIS…

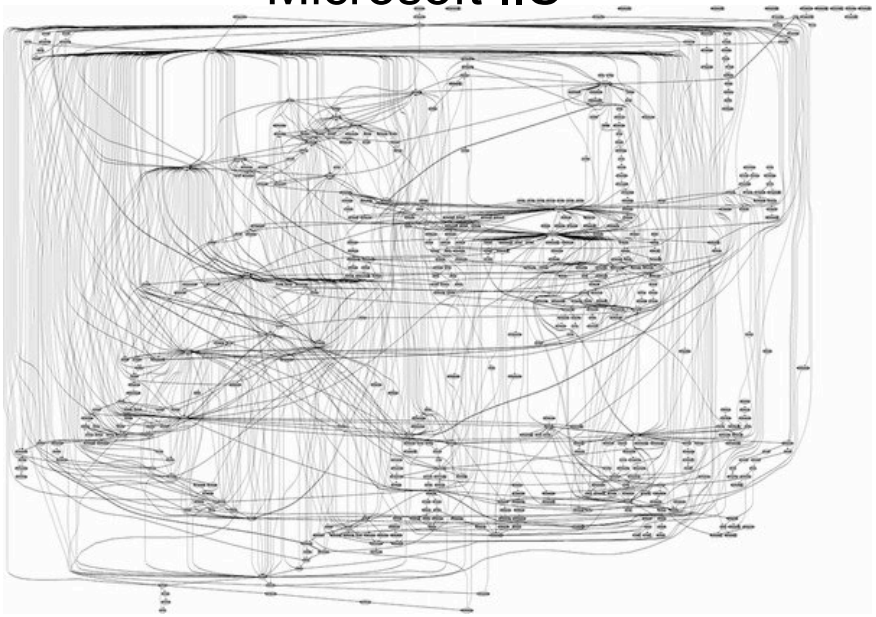


Enough Said!

Linux



Microsoft IIS



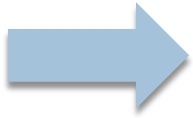
Lecture Outline

- Motivation for the AWAREness™ Business Blueprint.
- The AWAREness Business Blueprint™ (BB) specification
- Stakeholders of the AWAREness Business Blueprint™ (BB)

MOTIVATION FOR THE BUSINESS BLUEPRINT

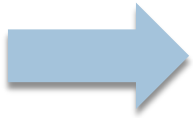
Motivation

- Software systems continue to grow in size and complexity



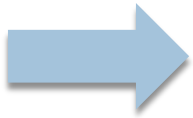
- Architectures offer level of abstraction to comprehend and analyze system vision.

- Software systems must meet stakeholders requirements



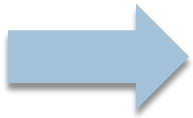
- Software architectures are useful artifacts for early system analysis. …
  - verification (traceability to requirements)
  - validation (correct realization of requirements in architecture)

- Stakeholder requirements change



- Architectures reflect decisions about how system will change (not) in response to stakeholder requirements change:

- While reuse is embraced, few applications can be reused with zero modification

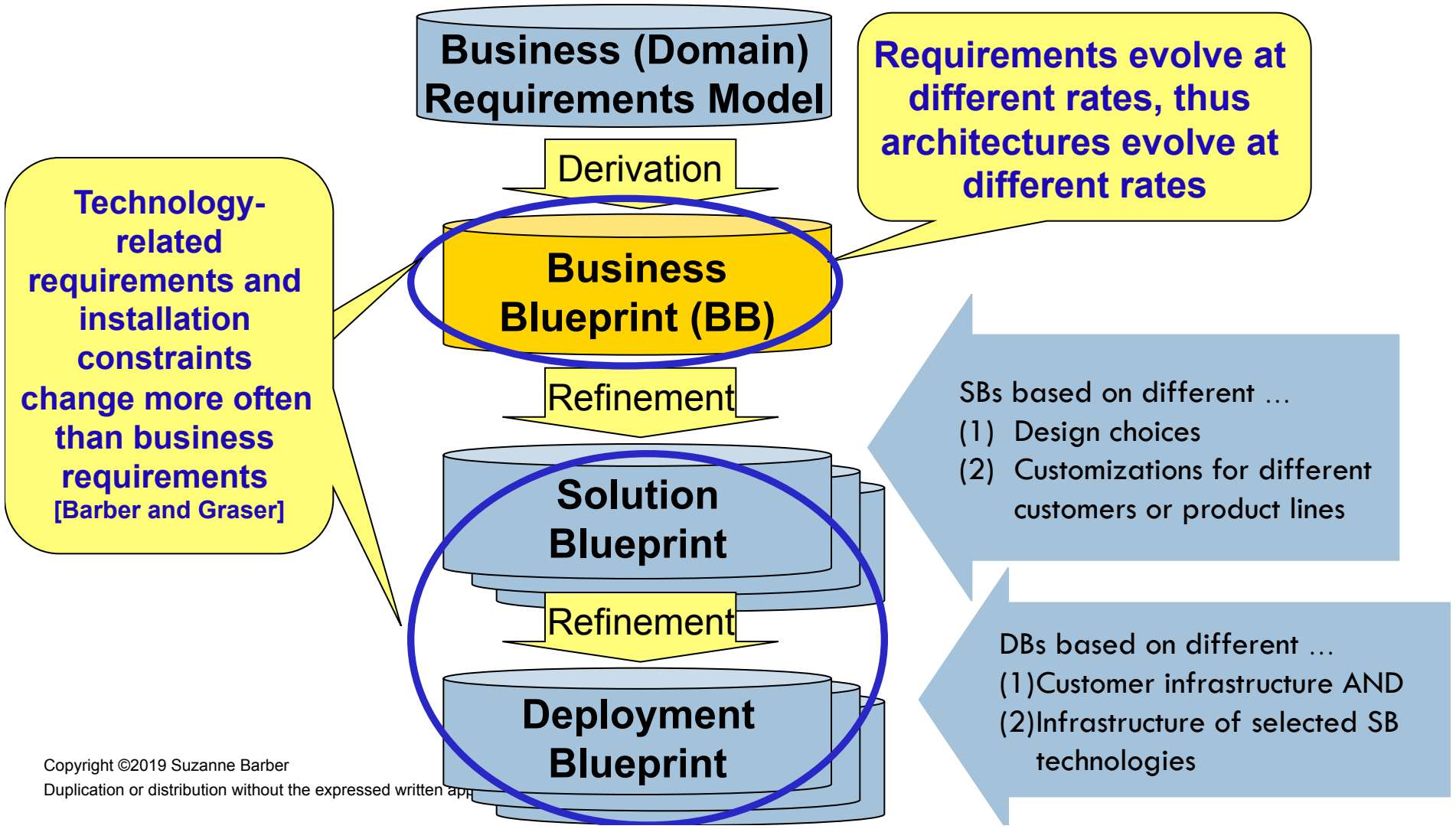


- Technology independent architecture can help identify candidate applications for reuse with respect to business functionality and data

Why create a Domain-Specific, Implementation-Independent Software Architectural View?

- Reuse, Reuse, Reuse of …
  - Domain (Business) Requirements: business functionality and data
  - Structural Requirements: relationships between components to which functionality and data have been assigned
- Change Management
  - Based on the hypothesis that Business Requirements change less often than Technology Requirements, this Business Blueprint is often more stable and changes less often.
- Comprehensibility
  - More stakeholders speak the "language" of the Business Blueprint --- Business.

AWAREness 3d Architecture™ specifications separate concerns across three Blueprints views.



THE AWARENESS 3D ARCHITECTURE: BUSINESS BLUEPRINT ARTIFACT

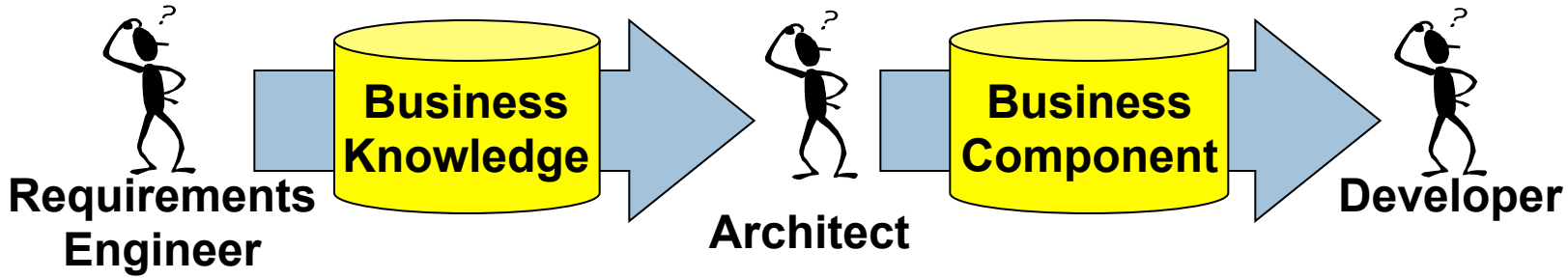
The AWAREness Business Blueprint serves as a Reference Architecture.

The Business Blueprint is a modular, technology independent view represented as related business components intended for reuse in building a "family" of Solution Blueprints (designs).

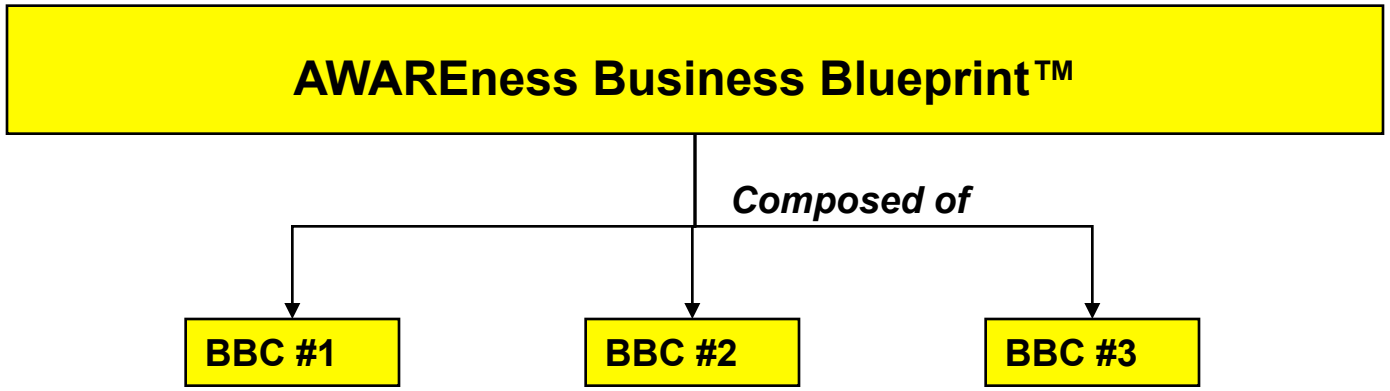
What are the Business Blueprint components?

In a more general sense, how are the components of any system decided?

- When performing analysis and design for an object-oriented program, the developer doesn't know the object classes a priori.
  - Want to define program classes to yield benefits such as reusability and maintainability.
    - *When you develop a C++ program, how do you decide what the objects will be? Does your organization have a manual?*
- Abstract this problem to the architecture derivation activity earlier in the lifecycle.
  - Want to define architecture component of business knowledge to provide a reusable, maintainable "blueprint" for developers.
    - *Very similar to problem above without implementation detail*



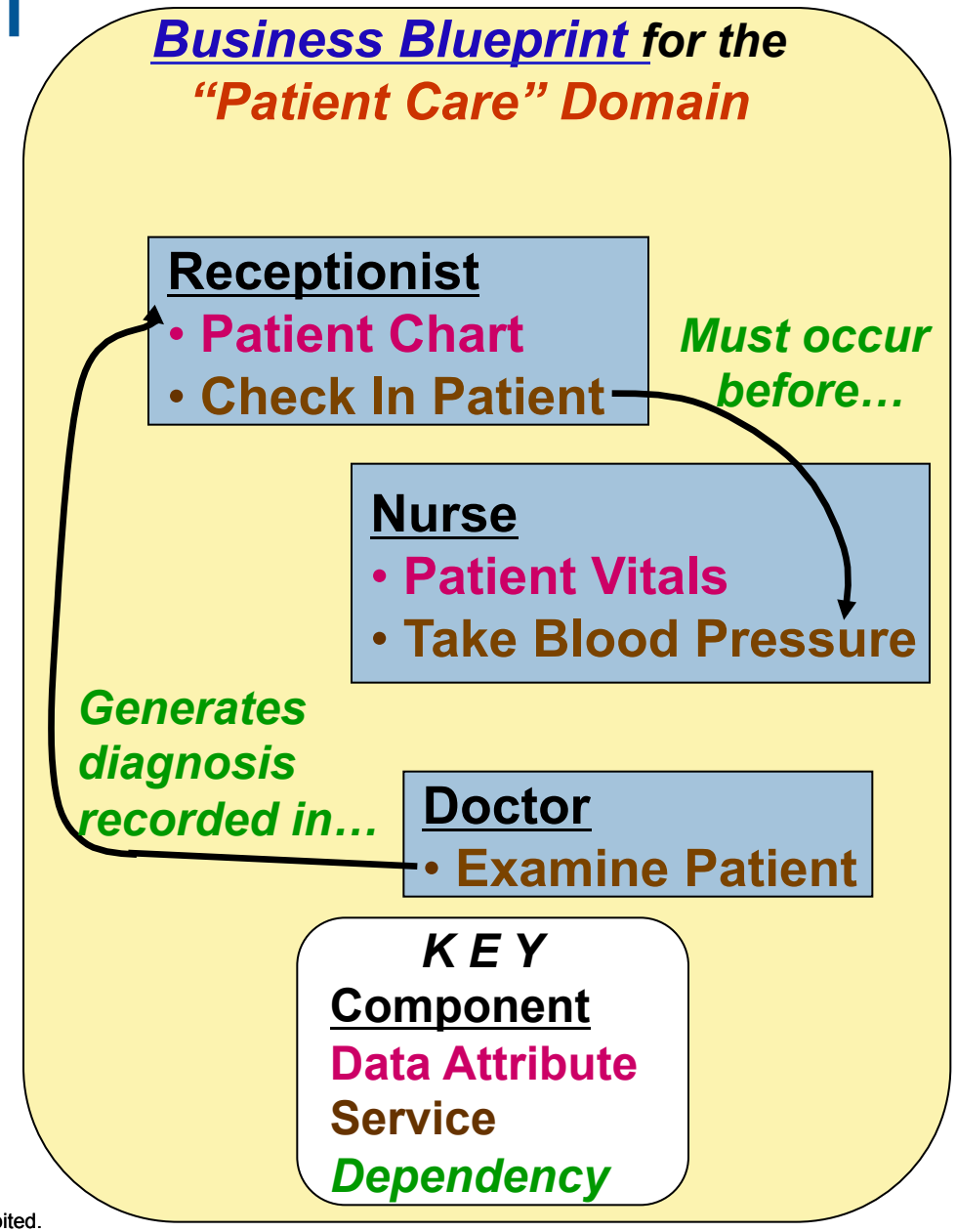
Business Blueprint Components (BBCs) constitute the Business Blueprint.



- Each Business Component should uniquely own service(s) and data
  - Business Component services = business tasks (capabilities) = functional requirements
    - BBC service specs include timing and "interface" requirements
  - Business Component attributes = business data = data requirements
- Tasks and data in Business (domain) Model assigned to respective Business Components

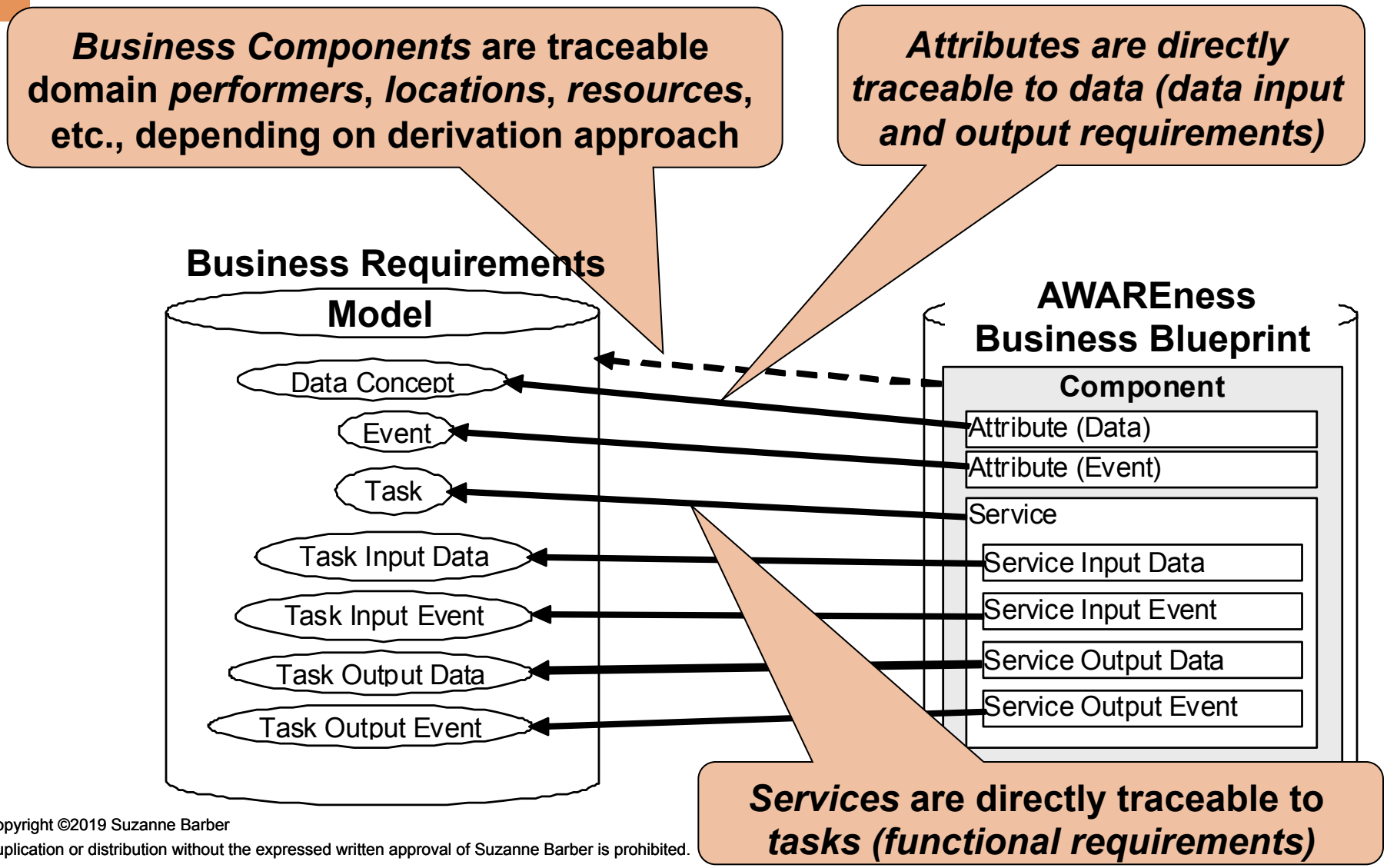
Business Blueprint

- AWAREness Business Blueprint™ is an architectural view to:
  - Provide a "blueprint" for developing a family of solutions in a domain
  - Capture business (functional, data, and timing) requirements
  - Serve as an Implementation Independent view of the system
  - Expresses an architect's vision by prescribing systems reflecting selected qualities

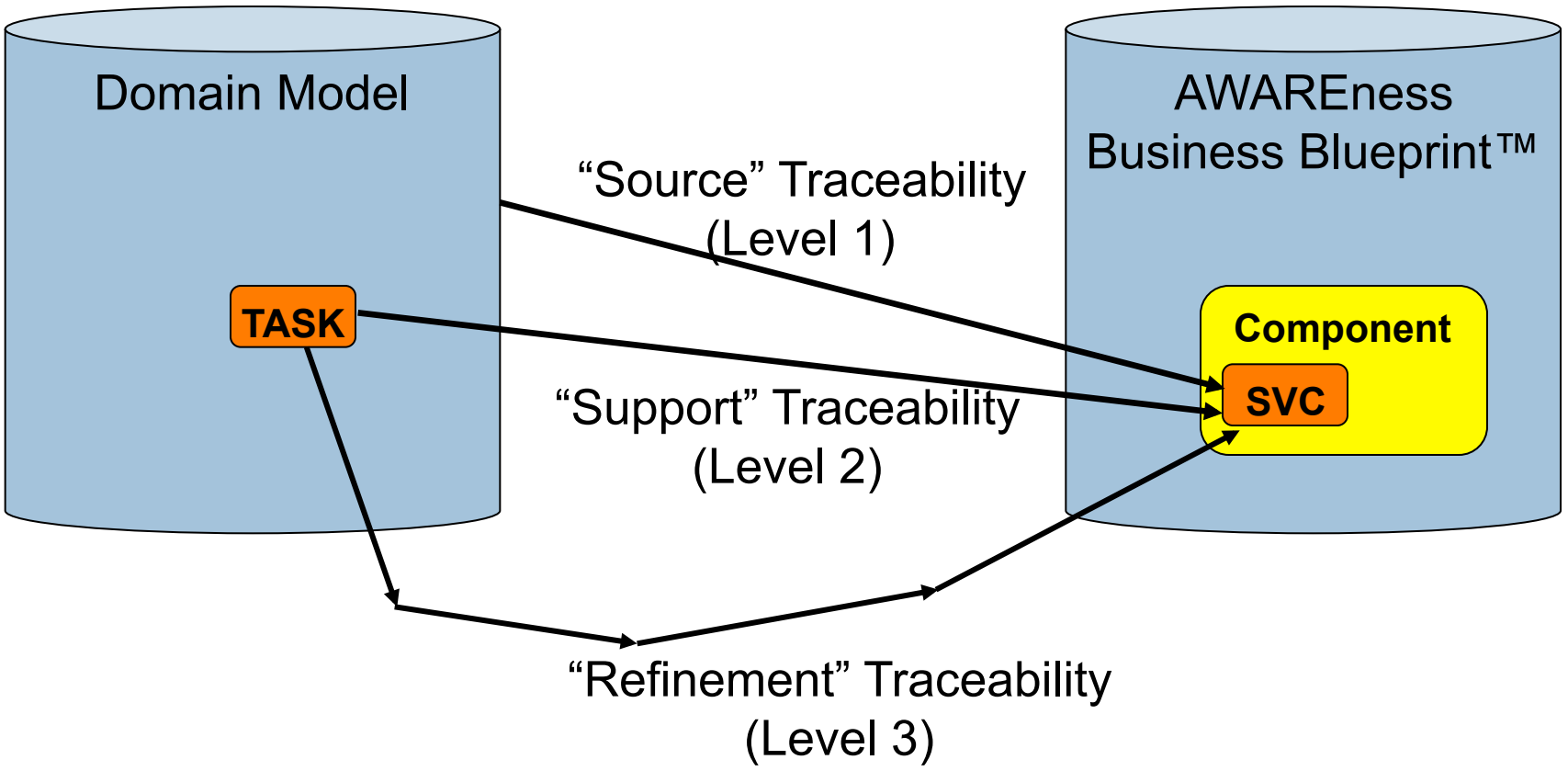


TRACEABILITY

Traceability – Architectural Business components traceable to elements Requirements Specification



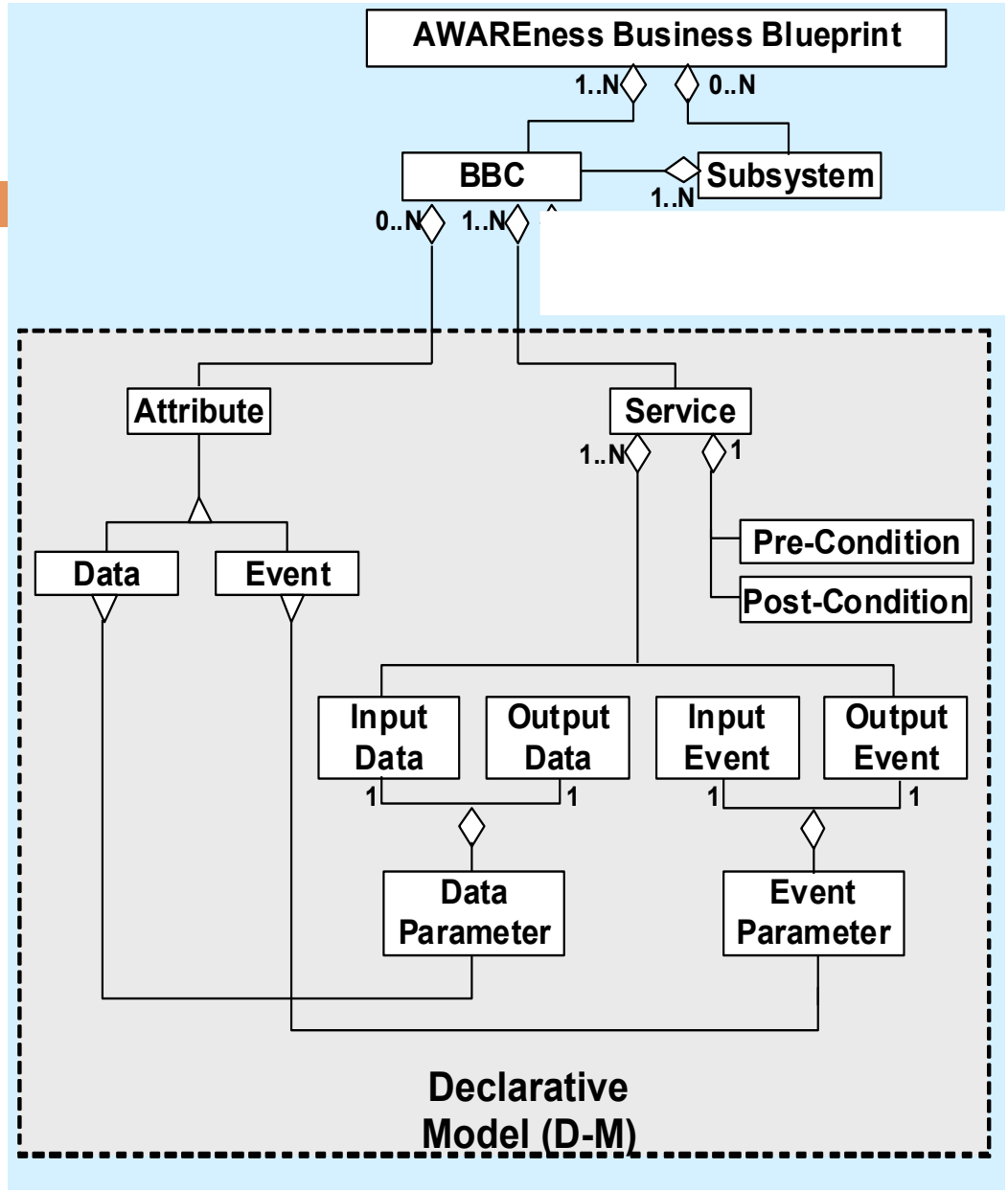
Traceability Representations



Announcements

- Project Milestone #1
  - Due Date extended to Today at midnight
  - Include a summary of who did what
    - Insert description in comments section of the assignment submission page on canvas.

- Quiz #1
  - Graduate Students: Average -- 6.2
  - Undergraduate Students: Average -- 5.8
  - Remember: Quiz will be best out of 2 quizzes
  - Policy for Regrade Considerations: Submit hardcopy of test and description of why you believe regrade is necessary (description written on test or on a separate attached sheet) submitted to Dr. B by Wednesday, October 9th, class time
    - You can submit to TA in office hours if necessary



- Every BB includes 1 or more BBC
- Every BBC has …
  - 1 to N Services
  - 0 to N Attributes
- Every attributes is a kind of …
  - Attribute
  - Data or Event I/O Parameter
- Every BBC Service has …
  - 1 PreCond statement and 1 PostCond statement (conditions combined as AND/OR expression)
  - 1 to N data or event input elements
  - 1 to N data or event output elements

Business Blueprint Component (BBC) Representation

AWAREness Business Blueprint Component™ (BBC)

Declarative Model (D-M)

Attributes

- Name : name of attribute
- Type : data type of attribute
- Cardinality: attribute data card.
- Value Constraints : expression

Services

- Name : name of service
- Preconditions: expression
- Postconditions: expression
- Input Events
  - Received from BBC service
- Input Data
  - Received from BBC service
  - Received from BBC attribute
- Output Events
  - Sent to BBC service(s)
- Output Data
  - Sent to BBC service(s)

Behavioral Model (B-M)

State Chart

- States : high-level states
- Transitions: high-level transitions
- Events:transition enabling events
- Guardstransition enabling guards

Integration Model (I-M)

Subsystem Dependencies

- Business Components:
  - elements of subsystem

Service Dependencies

- Business Component Services:
  - required service events generated from other Business Components

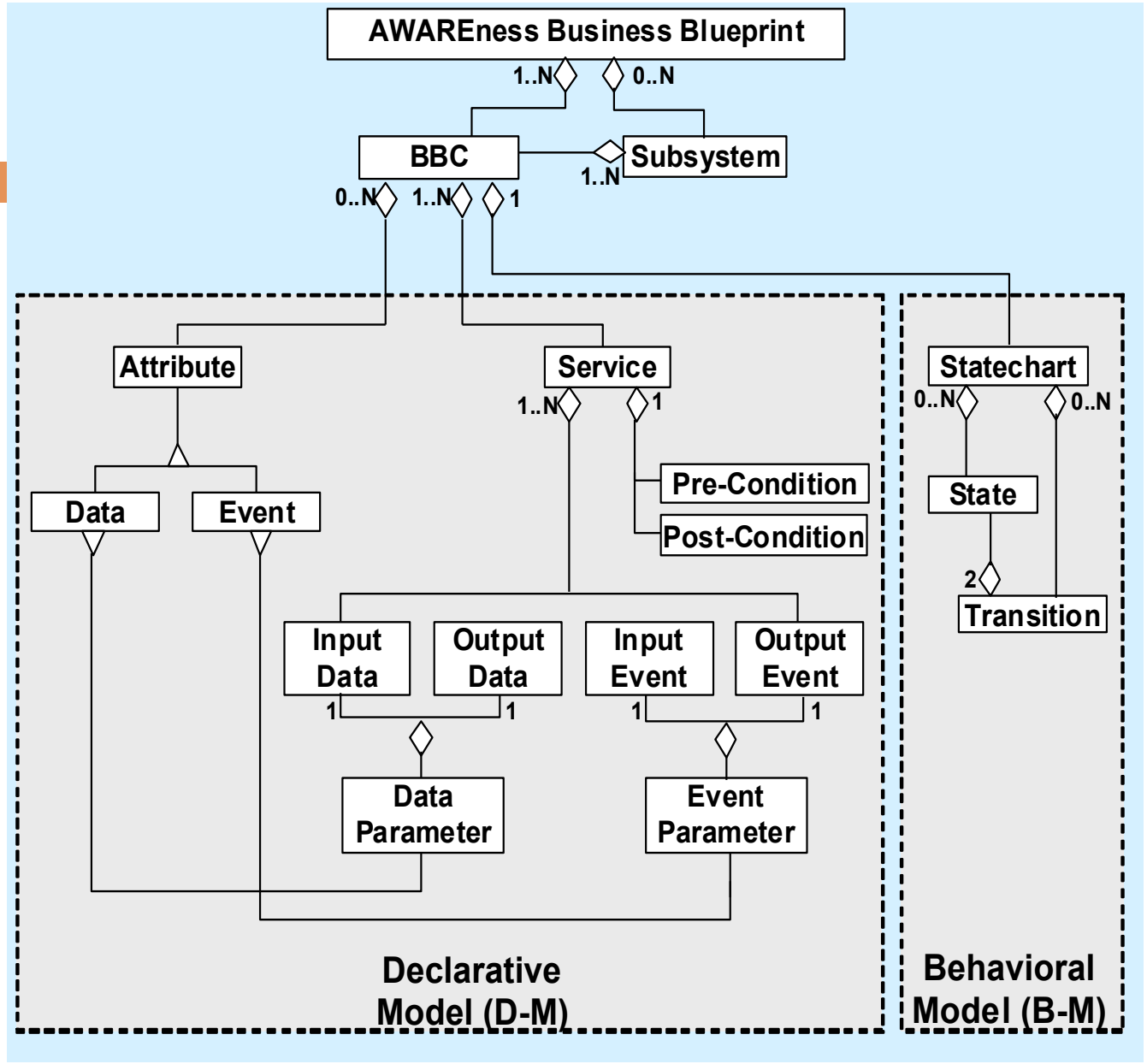
Attribute Dependencies

- Business Component Attributes:
  - required attributes

Business Blueprint Component Example

AWAREness Business Blueprint Component™ (BBC)

Declarative Model (D-M)	**Behavioral Model (B-M)**	**Integration Model (I-M)**
Attributes Name: *Patient Vitals* Type: *composite* Cardinality: 0-N Value Constraints: None	State Chart States - *Idle* - *Performing 'TakeVitals'* - *'TakeVitals' Error*	Subsystem Dependencies BBCs: None
Services Name: *Take Vitals* Input Events Event: *PatientCheckedIn* Recd From BBC: *Receptionist* Recd From Svc: *Check In Patient* <u>Output Data</u> Data: *Patient Vitals* Sent to BBC: *Doctor* Sent to Svc: *Diagnose Symptoms*	Transitions - From: *Idle* To: *Performing 'Take Vitals'* Guard: (Evt *PatientCheckedIn* recd from svc *Check In Patient* under BBC *Receptionist*) - From: *Performing 'Take Vitals'* To: *Idle* Guard: (Data *Patient Vitals* sent to svc *Diagnose Symptoms* under BBC *Doctor*) - From: *Performing 'Take Vitals'* To: *'TakeVitals' Error* Guard: ~(Data *Patient Vitals* sent to svc *Diagnose Symptoms* under BBC *Doctor*) - From: *'Take Vitals' Error* To: *Idle* Guard: (Evt *TakeVitals.Error-Complete* generated)	Service Dependencies BBC Services: - Svc *Check In Patient* under BBC *Receptionist* - Svc *Diagnose Symptoms* under BBC *Doctor* Attribute Dependencies BBC Attributes: None

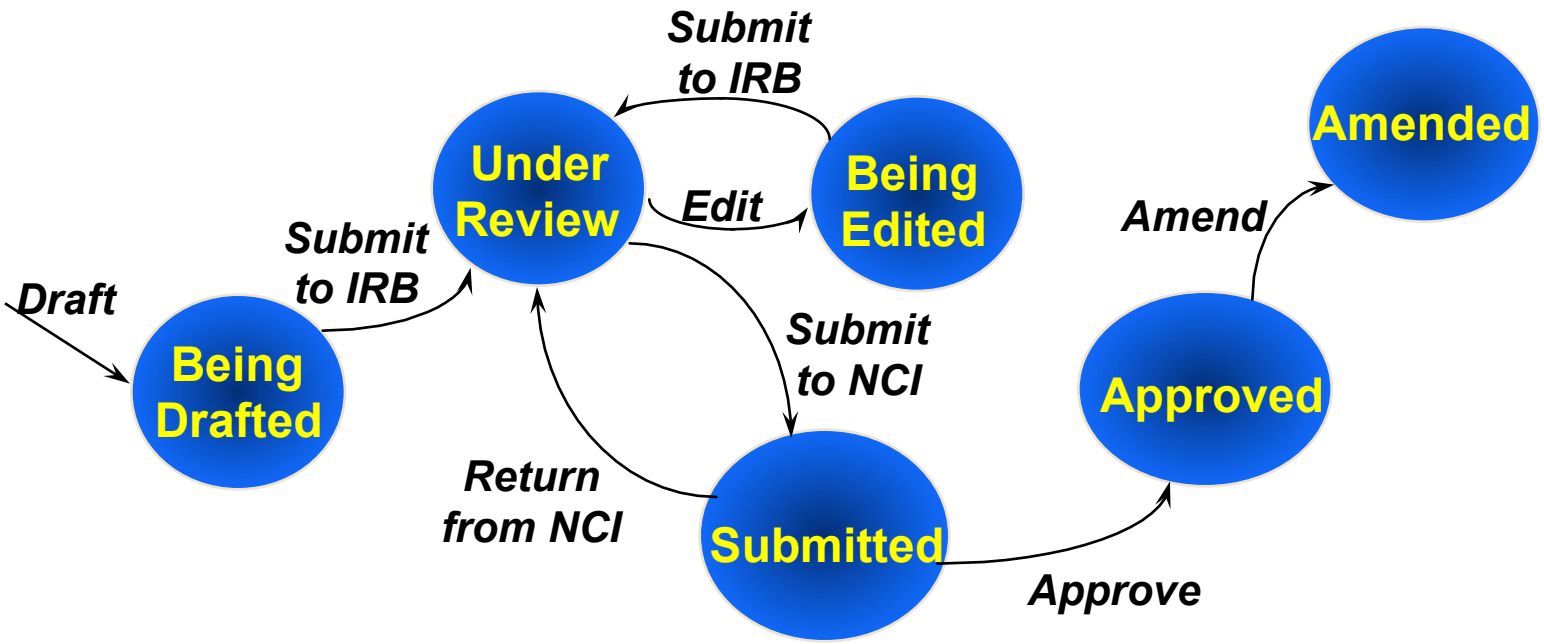


Business Blueprint: Declarative Model

- Component Specification Template:
  - Name
  - Classification
  - Attributes Required/Provided
  - Services Required/Provided
- Example: *NCI Protocol*
  - *Name:* NCI Protocol
  - *Classification/Inheritance:* NCI Component
  - *Attributes:* Eligibility, Treatment
  - *Services:* Draft, Submit, Amend

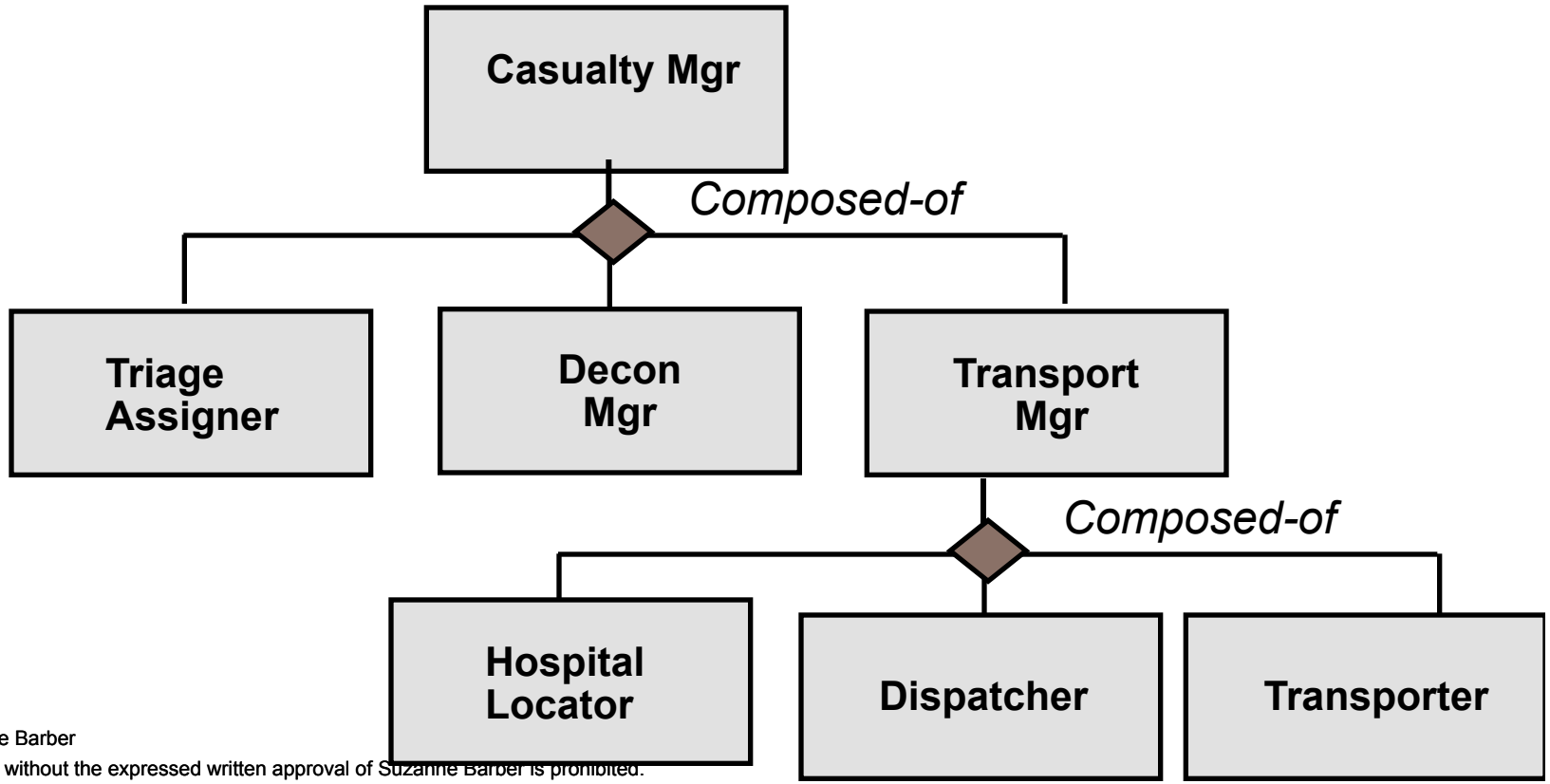
Business Blueprint: Behavioral Model

- BB Behavioral Model describes the states of the BBC and the transitions (conditions) to move from one state to another.
- Example: *NCI Protocol*

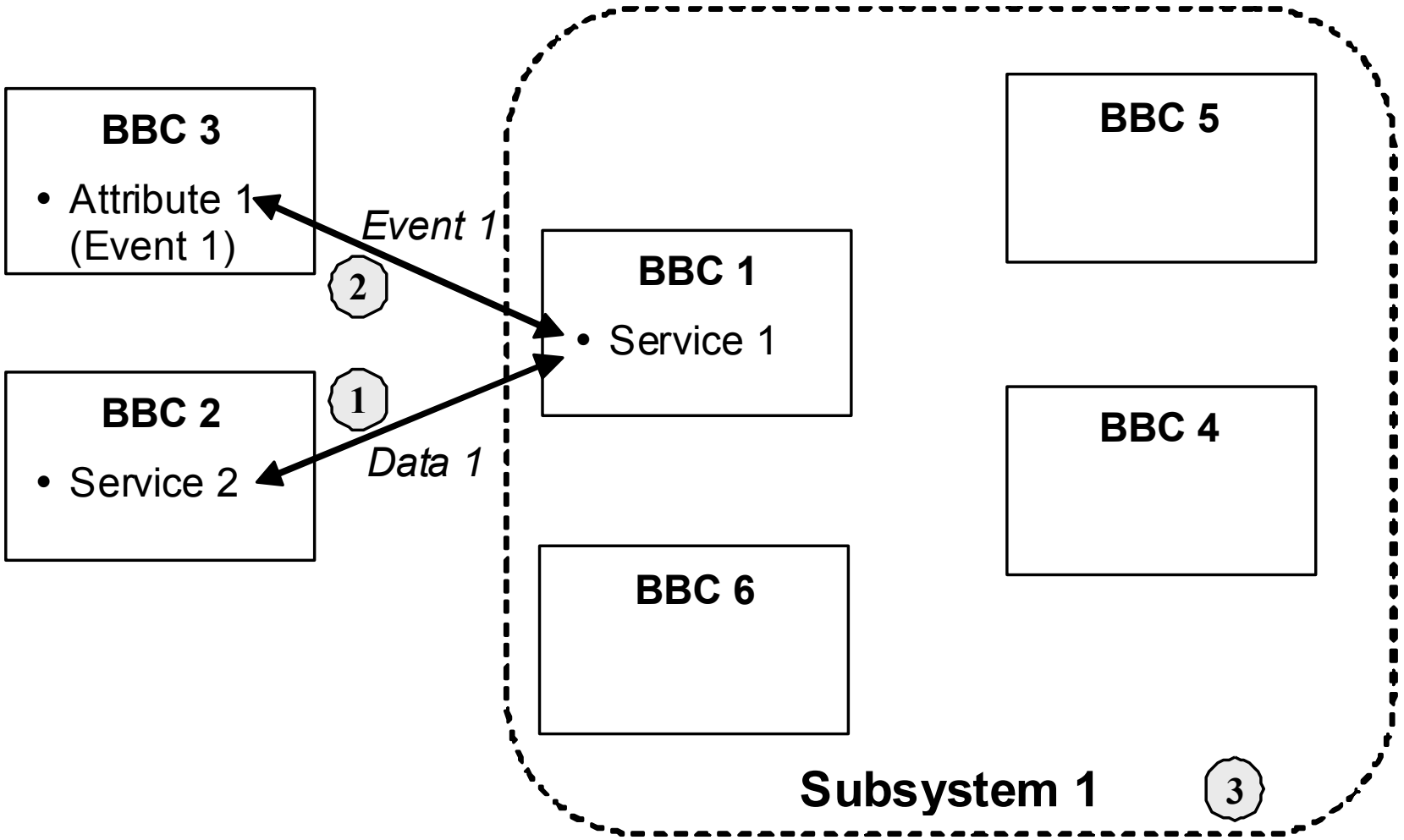


Business Blueprint Integration Model of Subsystems

- Collections of Business Components into Subsystems often result from
  - Identified high levels of Business Component service dependencies
  - Common practices (legacy systems) found in the domain

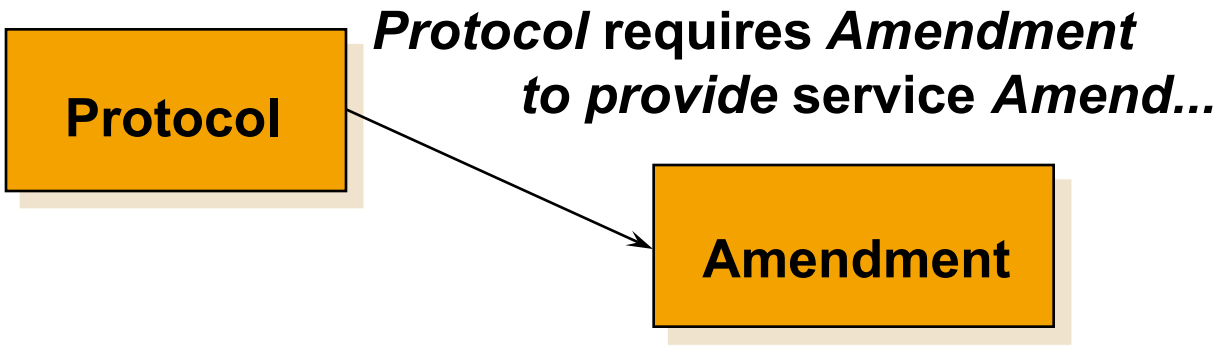


BB Integration Model: Business Component Dependencies represented in the Integration Model

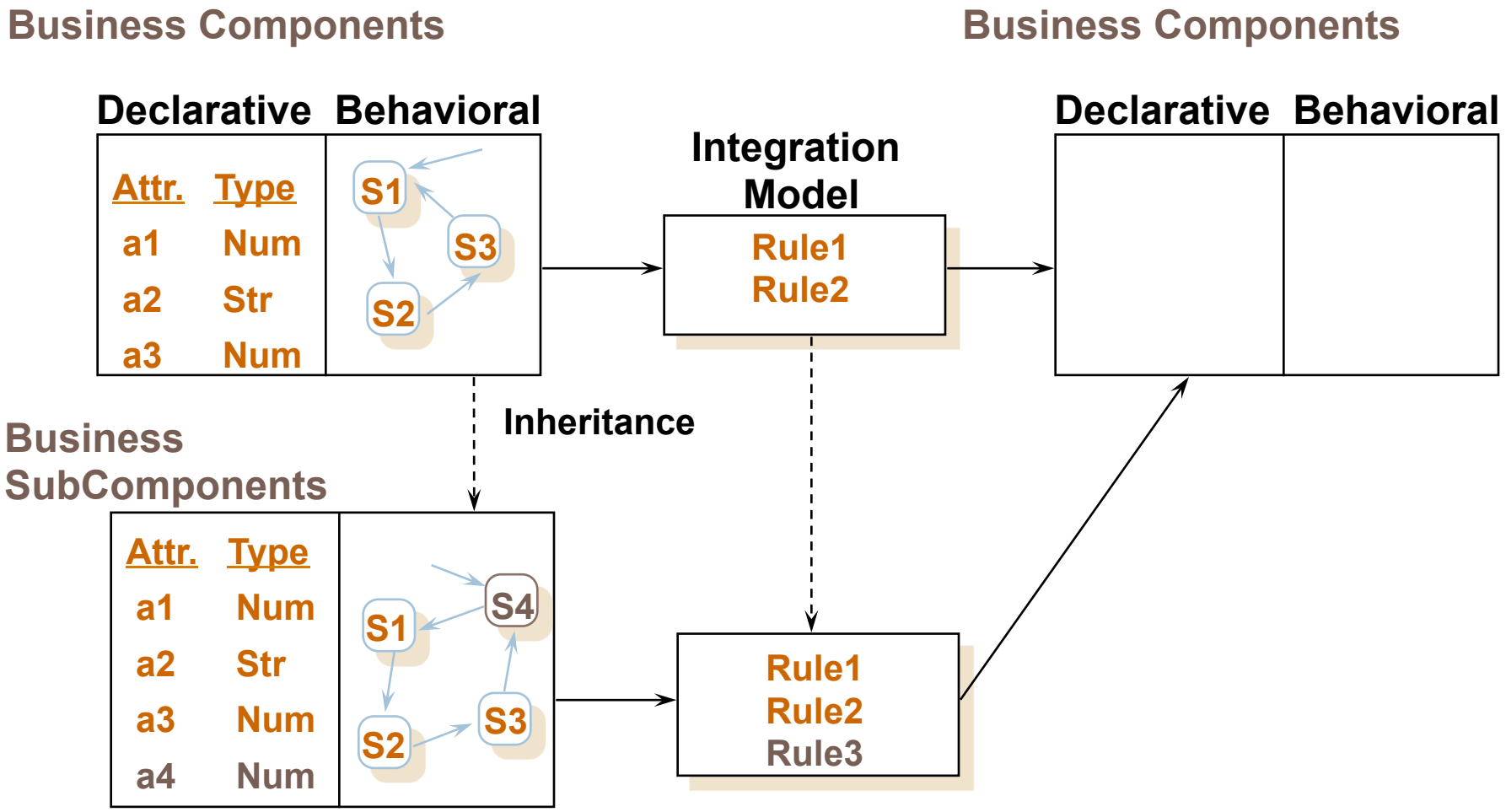


Business Blueprint: Integration Model

- Rules of Composition:
  - Set of constraints and dependencies between components.
- Example: *NCI Protocol*
  - IF need to *Amend*
  - THEN *Protocol* must have access to *Amendment* component

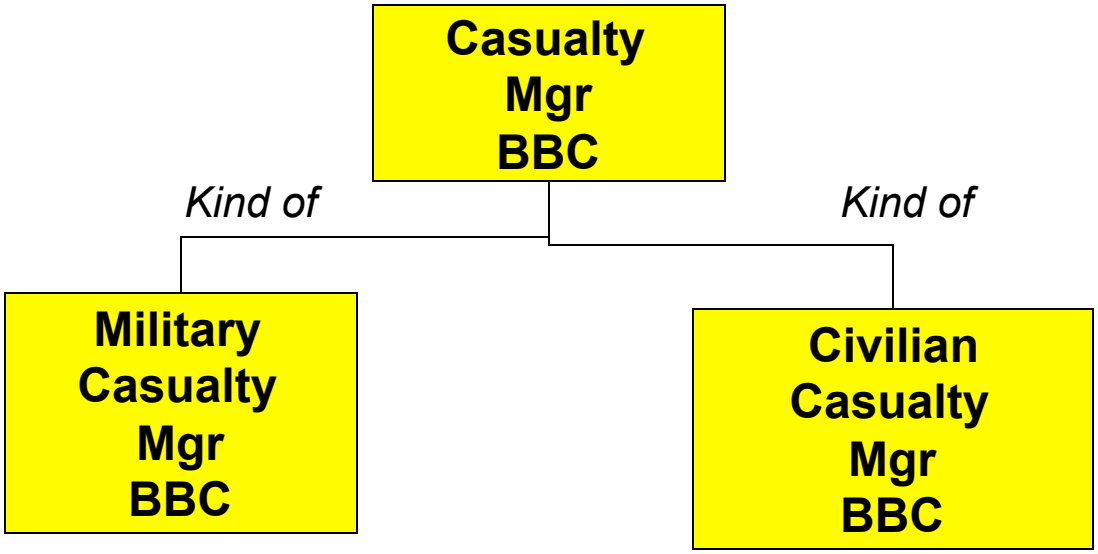


Specialization of Business Blueprint Components

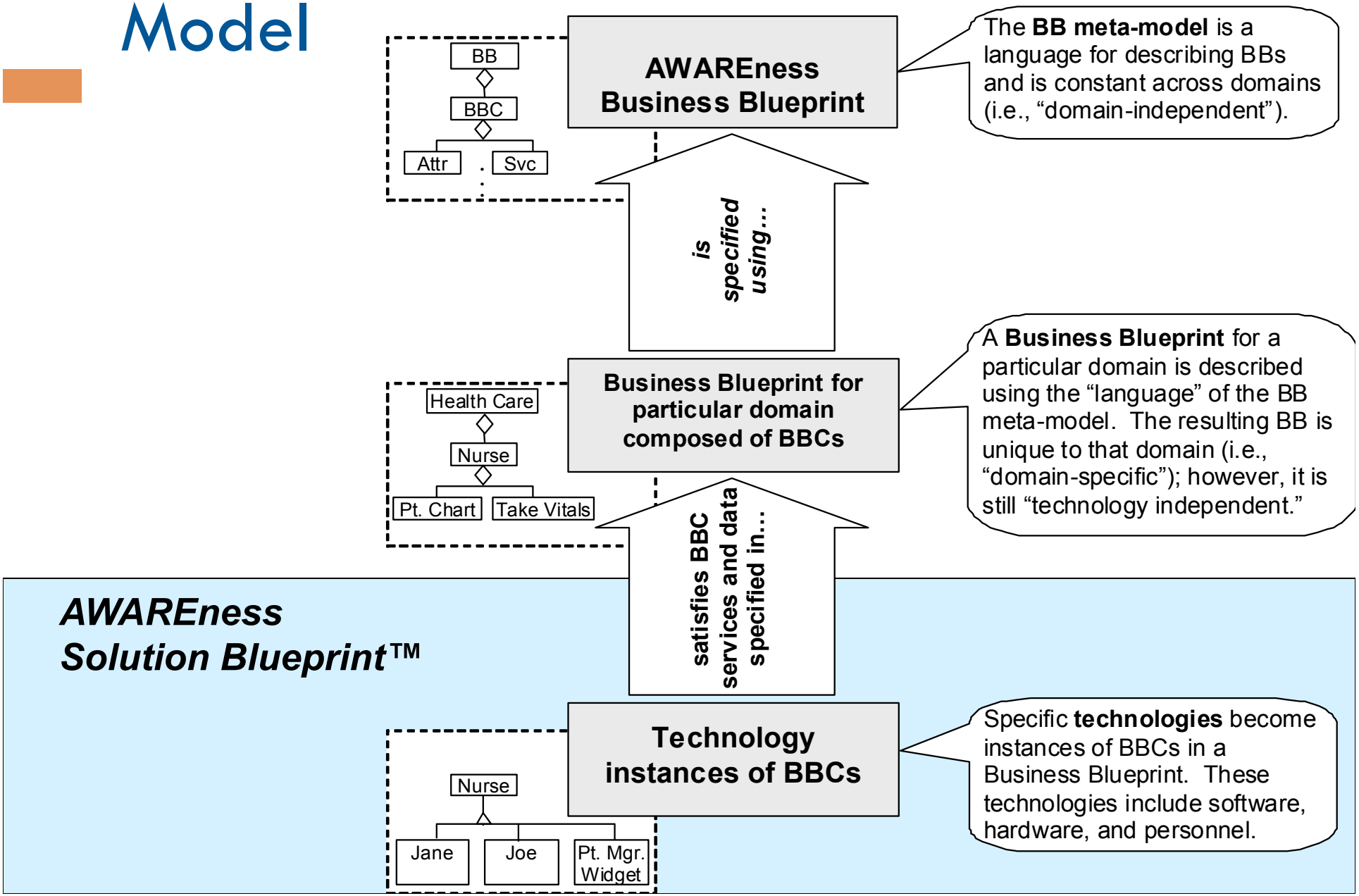


Business Blueprint Component Hierarchies

- Constructed based on common service and data/event responsibility assignments



AWAREness Business Blueprint™ Meta-Model



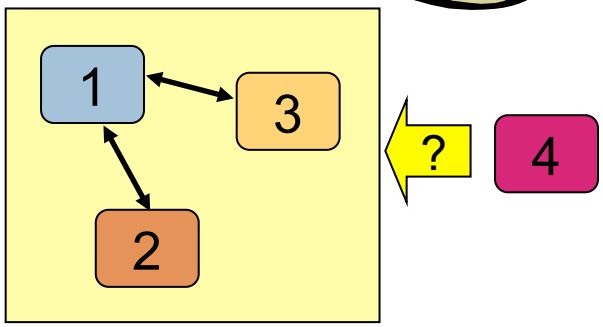
Who are "customers" of the Business Blueprint?

Developers

Implementation independent specification

System Integrators

Interactions between components to guide system configuration and testing

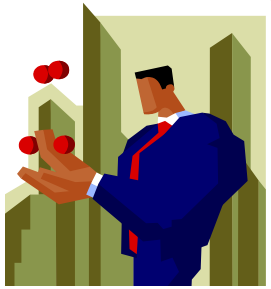


Knowledge Engineers

Barometer on completeness of domain knowledge

Project Managers

Assignment of domain requirements to developers



Back to the Cub Scout Training Registration System -- .

- If you received requirements to include the following functionality and data, draw a box and line architecture representing the Business Blueprint components.

- **Functionality**
  1. Collect Scout name and Troop #
  2. Collect requested courses
  3. Check course availability
  4. Assign instructors to courses
  5. Schedule course dates and times
  6. Schedule course location
  7. Collect course resources
  8. Pay for course resources
  9. Assign Scout credit for courses
  10. Send Scout badges for earned course credits

- **Data**
  - Scout Name
  - Troop #
  - Course Name
  - Badge Type
  - Scout course credits
  - Course payment
  - Resource fees
  - Course location
  - Course date
  - Course time

- How do you know these requirements are the "right" ones? And at the "right" level of abstraction?
- How do you know this is the correct architectural structure? In other words,
  - the "right" allocation of responsibilities for functionality and data?
  - The "right" vision for the system design and implementation?
- What additional information would have been helpful to make better decisions about allocation functionality and data to components?