ARCHITECTURAL STYLES (PATTERNS)

Dieter K. Hammer

Department of Computing Science
Eindhoven University of Technology (EUT)
Eindhoven, The Netherlands

E-Mail: d.k.hammer@tue.nl
WWW: http://www.win.tue.nl/~hammer

PATTERNS IN ARCHITECTING

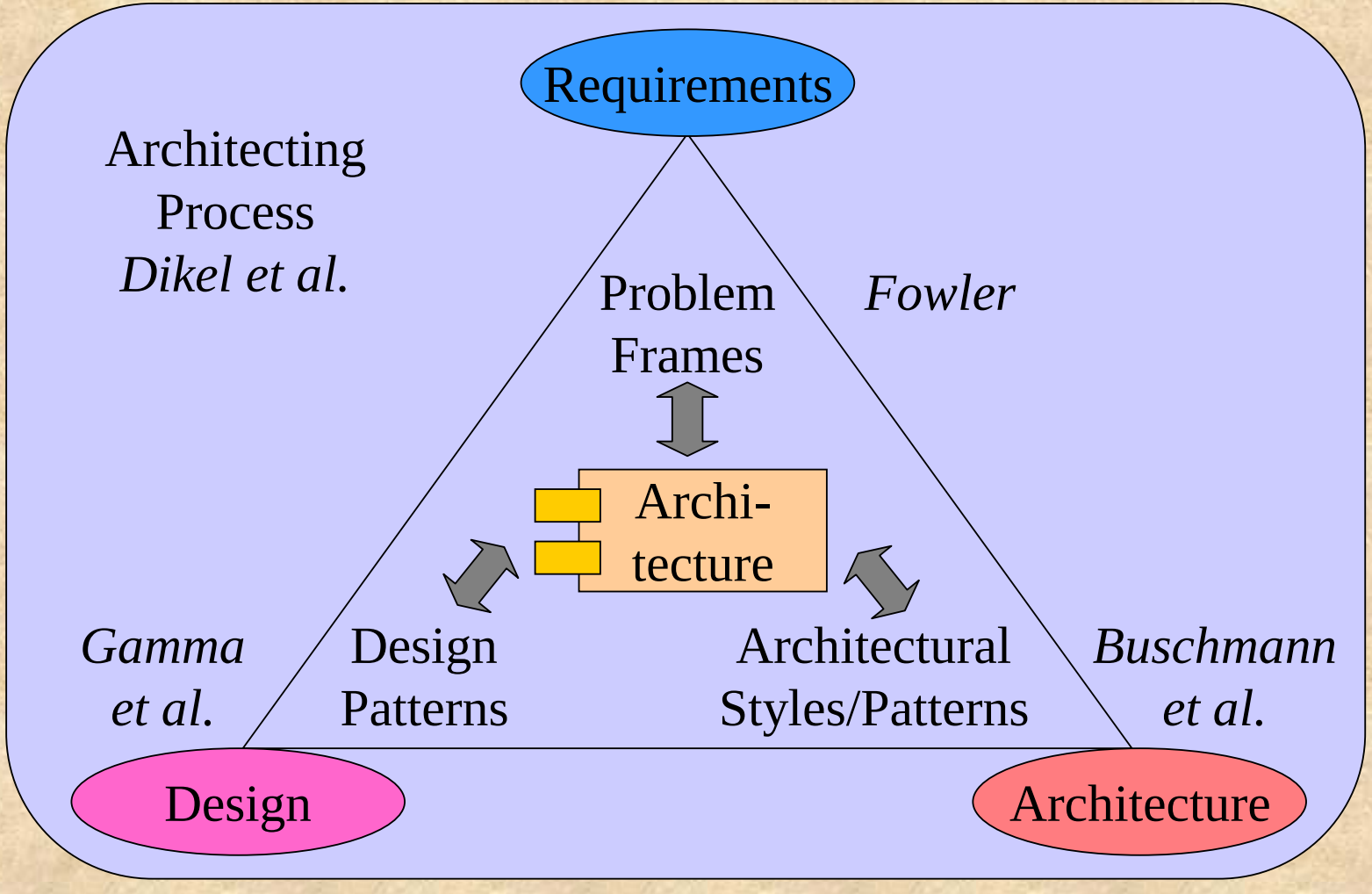
- **Christopher Alexander** (The timeless way of Building, 1979) defined spatial patterns that support events that frequently happen in a certain place by solving the conflicting forces of interest in a generic way
- These patterns are hierarchically linked (town, quarter, house, room) and form a **pattern language**
- They are related to the living quality of the people involved and support the **timeless quality without a name** that cannot be reduced to a single dimension

PATTERNS IN SE

Software Eng. patterns are much more restricted:

- They support merely the engineer and not the user
- The relation with architectural qualities is weak
- This holds especially for human-related QA's like usability, touch & feel, etc.
- The collection and the linking between individual patterns are not complete enough to form a language
- Many patterns are workarounds to implement shortcomings of OO languages like C++

PATTERN-DRIVEN DEVELOPMENT



PATTERN TYPES

	Problem Frames	Architectural Styles/Patterns	Design Patterns
Emphasis	Decomposition Separation of Concerns	Reuse Conceptual integrity	Reuse Conceptual integrity
Abstraction Level	Application Domain	Architecture	Component
Use	Concept that supports the analysis	Concept that needs to be adapted	Directly applicable

SOME LITERATURE

Martin Fowler, Analysis Patterns for Reusable Object Models, Prentice-Hall, Addison-Wesley, 1997

Erich Gamma, Richard Helm and Ralph Johnson, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

Frank Buschmann et al., Pattern-Oriented Software Architecture Vol. 1 & 2, Wiley, 1996 & 2000

David. M. Dikel, David Kane and James R. Wilson, Software Architecture: Organizational Principles and Patterns, Prentice-Hall, 2001

WHAT ARE ARCHITECTURAL STYLES?

- Architectural styles are *construction paradigms* for (a set of) design dimensions that describe system *patterns* and *characterize* a class of architectures
- Architectural styles are abstract
- New styles will emerge as the field develops
- Architectural styles embody best practices and make this knowledge applicable for future use
- Architectural styles are important as stable and proven solutions for variation points of the design

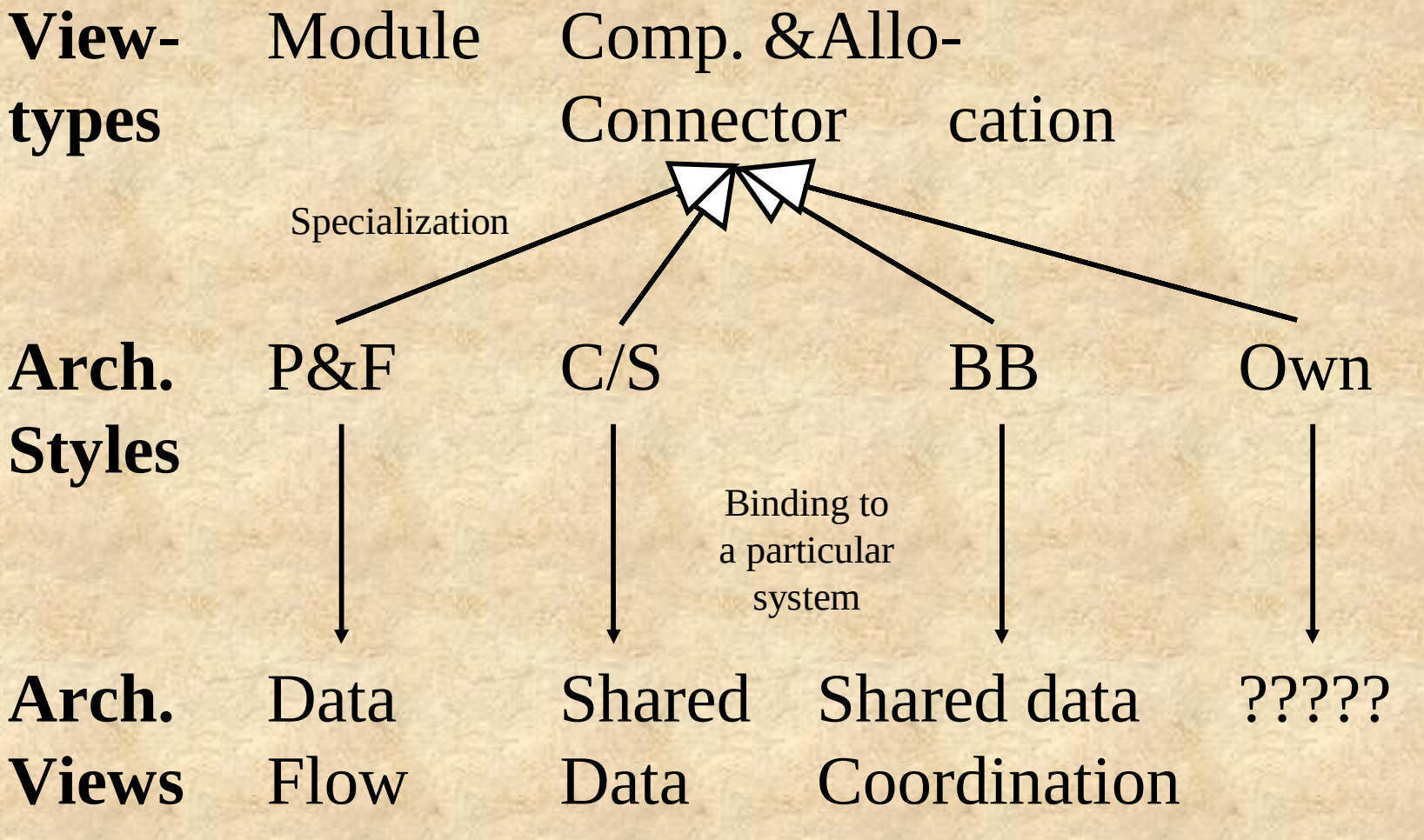
DESCRIPTION OF ARCH. STYLES

An architectural style is described by:

- A set of **concepts** that are defined in terms of:
  - **Motivation** and general description
  - **Functionality**: Set of component-, interface- and connector types
  - **Structure**: Static layout of these entities
  - **Interaction**: Run-time relationship between these entities
- A set of semantic **constraints**
  that preserve architectural integrity
- A set of **rules & guidelines**
  that support the application of the style

ARCH. STYLES DEFINE ALSO VIEWS

(From Clements et al., Documenting Software Architectures, 2003)



HETEROGENEOUS STYLES

An architecture can use several architectural styles and styles do not partition it into non-overlapping parts

Types of heterogeneity:

- **Locational**: Different areas of the system may exhibit different styles
- **Hierarchical**: An entity playing a part in one style uses sub-entities arranged in another style
- **Situational**: The system might be seen in different lights, depending on the viewpoint

INSUFFICIENT ATTENTION FOR QA's

Style descriptions mention QA's only sporadically

- No systematic description on how architectural styles influence quality attributes
- No systematic mappings from QA's to arch. styles

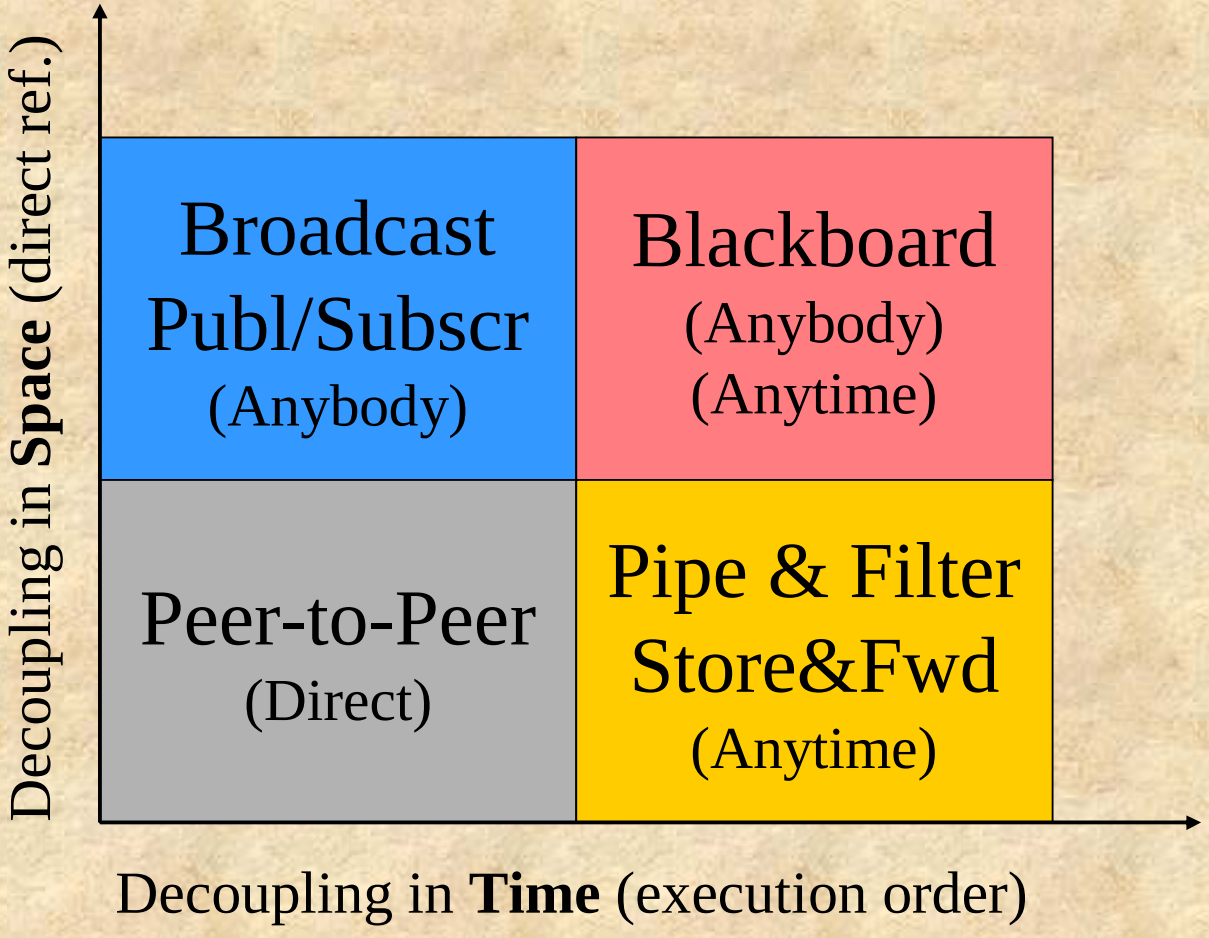
ABAS (Attribute-Based Arch. Styles):

- Pre-analyzed software architecture parts
- Rick Kazman and Mark Klein from CMU/SEI started to describe arch. styles that support certain QA's, incl. the relevant design & analysis models

ARCH. STYLES SUPPORT FLEXIBILITY

- Styles are proven solutions that solve design constraints in a generic - and thus flexible – way
- Properly used, they help architects to concentrate on the essential problems, i.e. to define only what is strictly necessary - and not more

COORDINATION PARADIGMS



EXAMPLE BLACKBOARD STYLE

Scalability: Transformations can be easily added

Flexibility: Functionality of clients is easy to change
+ Asynchronous: Anytime
+ Active BB: Anybody (P/S interaction)

Robustness: + Clients can be replicated,
- Blackboard is single point of failure in single processor implementation

Security: - All process share the same data
+ Security measures can be centralized around the blackboard