A Database Perspective on Building Large Applications

Joachim Thomas
UBS AG
4002 Basel - Switzerland

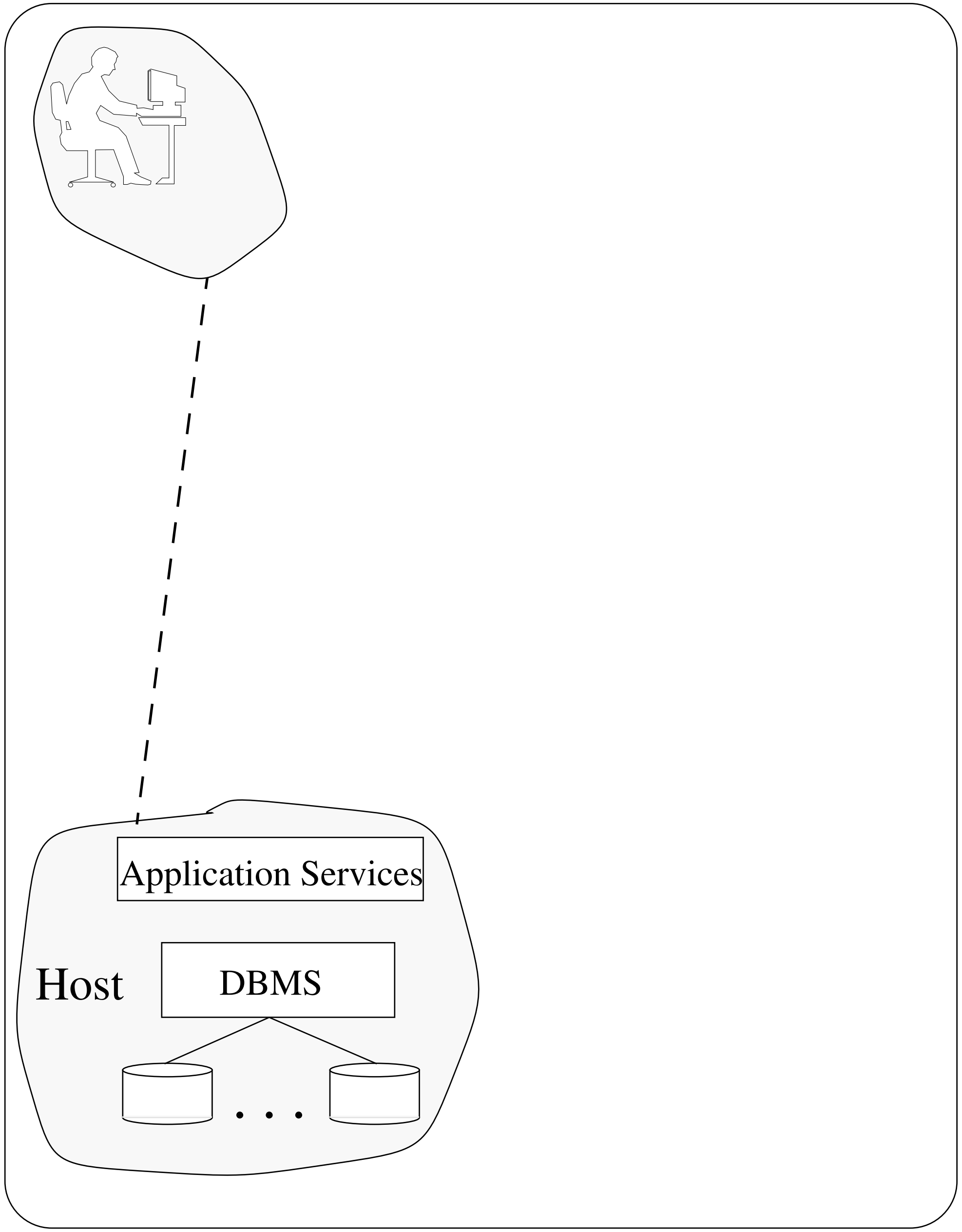
Overview

1. Historic Evolvement
2. Problems in Current Application Development
3. Choosing The Right Technologies
4. Conclusions

1. Historic Evolvement

Mid 1960's: Terminal applications

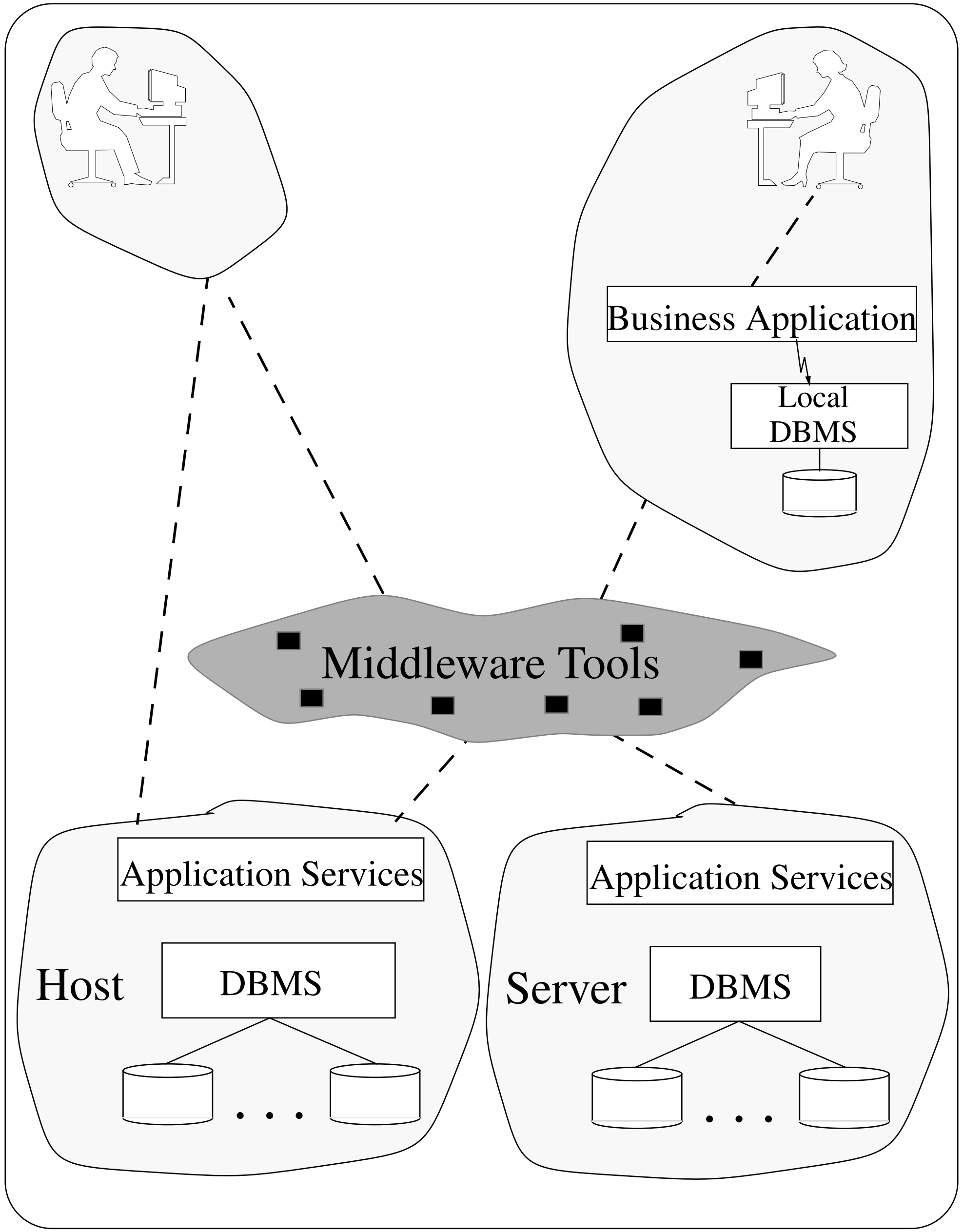
- Application logic entirely at the host.
- Client terminals serve I/O purposes only.
- Uniform, "slim" connections between clients and host.
- Centralization w.r.t.
  - Management and access of data,
  - Application development,
  - Application processing,
  - Application maintenance/evolution.



1. Historic Evolvement (cont.)

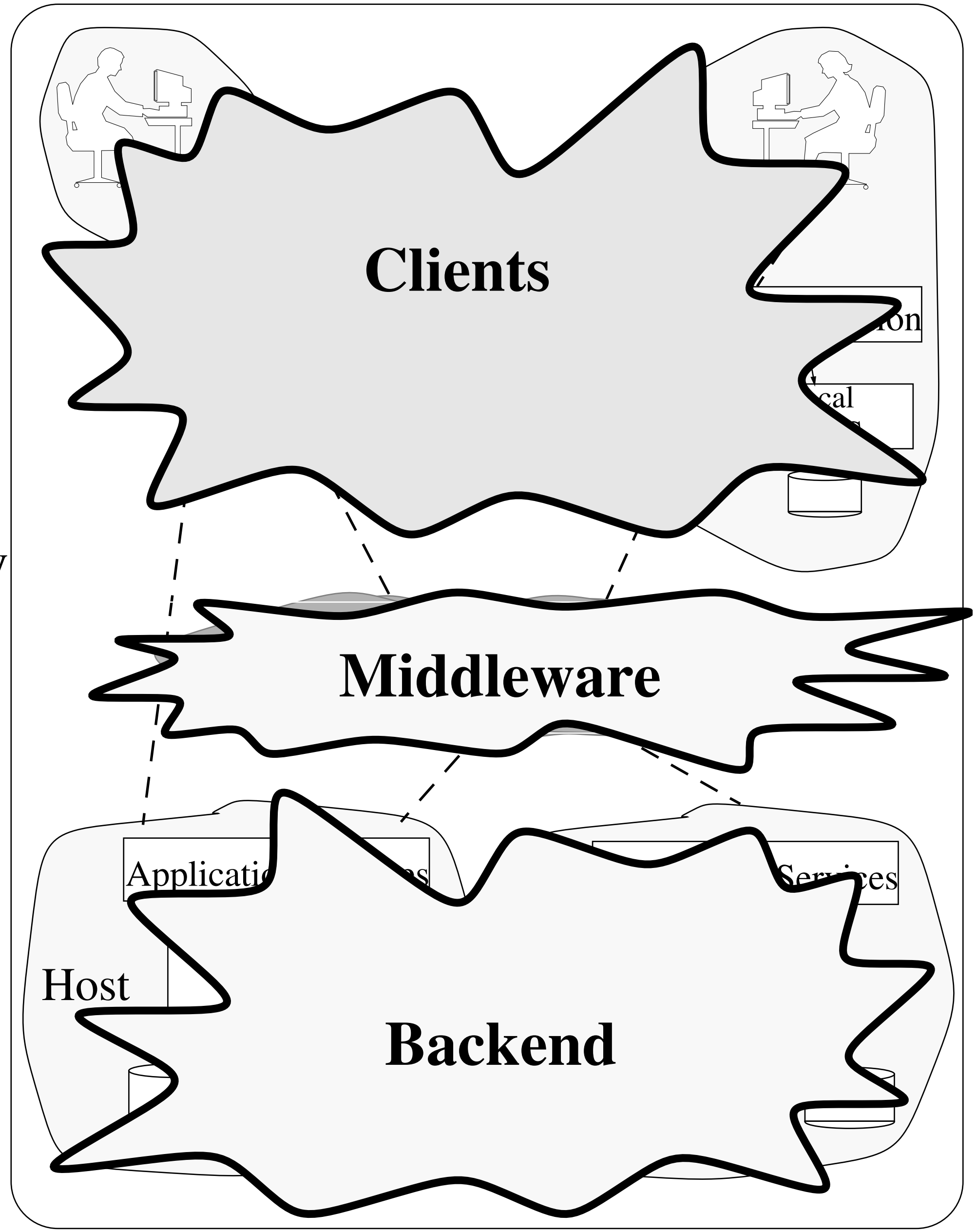
Over time: Gradual decentralization

- Application logic stretching from clients to host/server.
- Clients no longer uniform, “dumb” terminals.
- Diverse connections between clients and host/server, e.g.,
  - Communication software,
  - TP monitors,
  - Transformation tools,
  - Replication tools.



1. Historic Evolvement (cont.)

- Partial decentralization w.r.t.
  - Managment and access of data,
  - Application development,
  - Application processing,
  - Application maintenance/evolution.
- Three levels of processing, possibly embodying multiple tiers



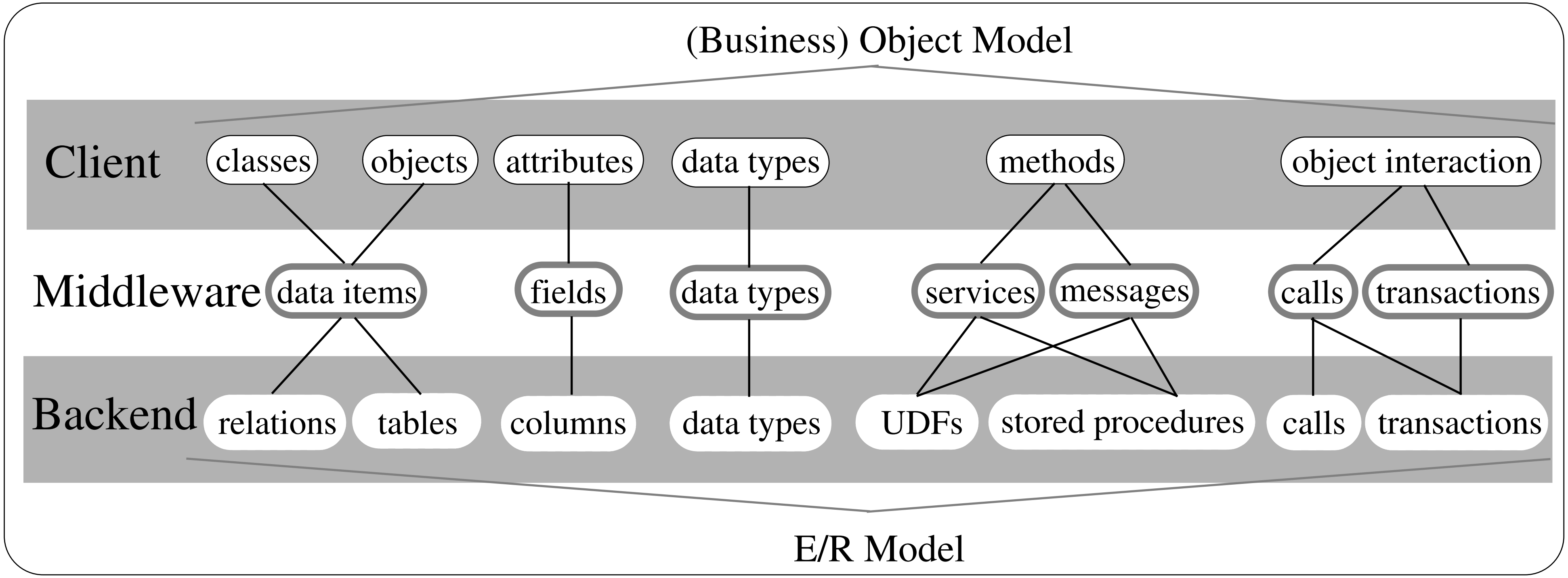
2. Problems in Current Application Development

- Different semantic notions, different levels of abstraction

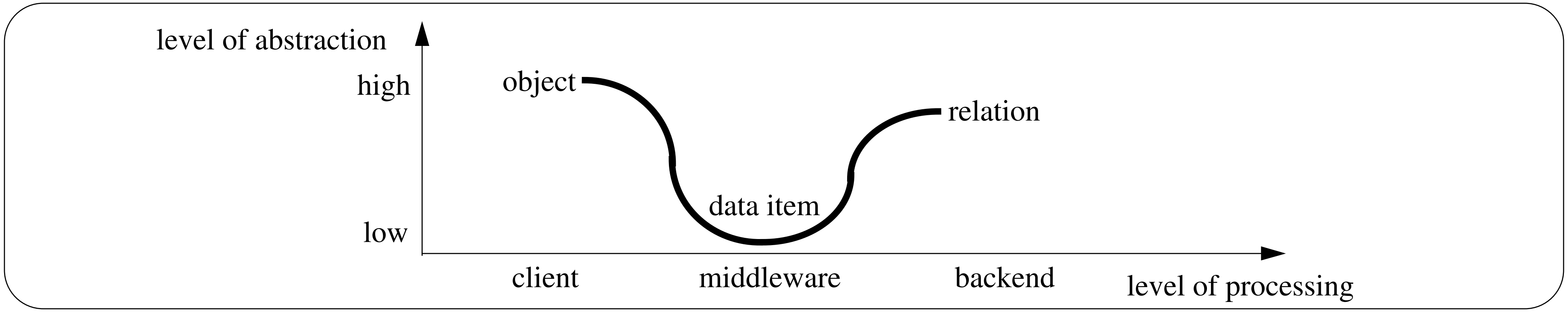
  ➨ vertically (along the path from clients to backend),

  ➨ horizontally (at each level of the path).

- Example: Client-based OO application (no OO middleware).



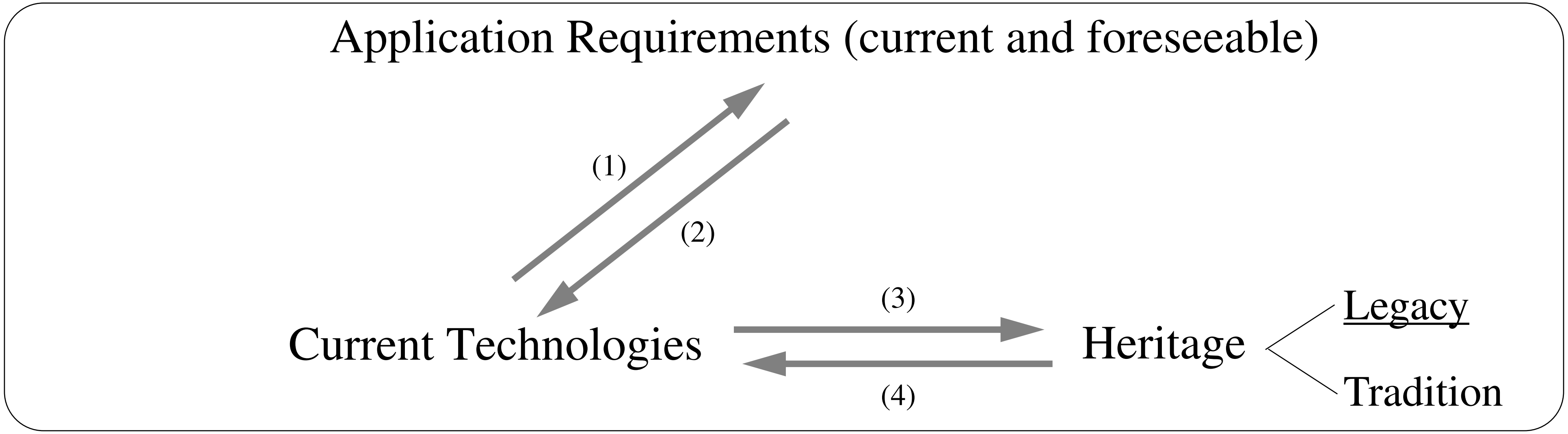
- Semantically close concepts defined and implemented at different levels (et v.v.)
  - ➨ No uniform application development, maintenance, and evolution.
- Many semantic gaps
  - ➨ Costly conversions, “semantic helter-skelter”.



- Disparate spheres of control and processing
  - ➨ Comprehensive transactional processing difficult to achieve,
  - ➨ No single point of authentication and authorization,
  - ➨ No uniform error reporting and error handling.

➨ Hard to achieve: Quality/reliability of design, implementation, maintenance.

3. Choosing The Right Technologies

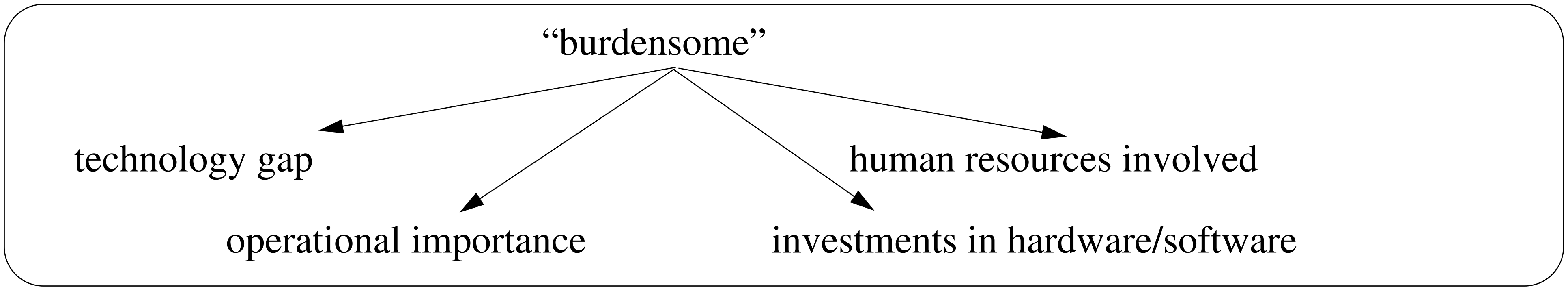


	Impact	Examples
(1)	Availability of new technologies	Internet, electronic commerce
(2)	Urgent application requirements	Security issues
(3)	Upgrading of existing solutions	Operating system (e.g., for legacy backends); object-relational technology
(4)	Exploiting heritage investments	Mapping tools, CORBA, EJB

3.1 Legacy

Free On-Line Dictionary of Computing:
"A computer system or application program which continues to be used because of the prohibitive cost of replacing or redesigning it and despite its poor competitiveness and compatibility with modern equivalents. The implication is that the system is large, monolithic and difficult to modify."

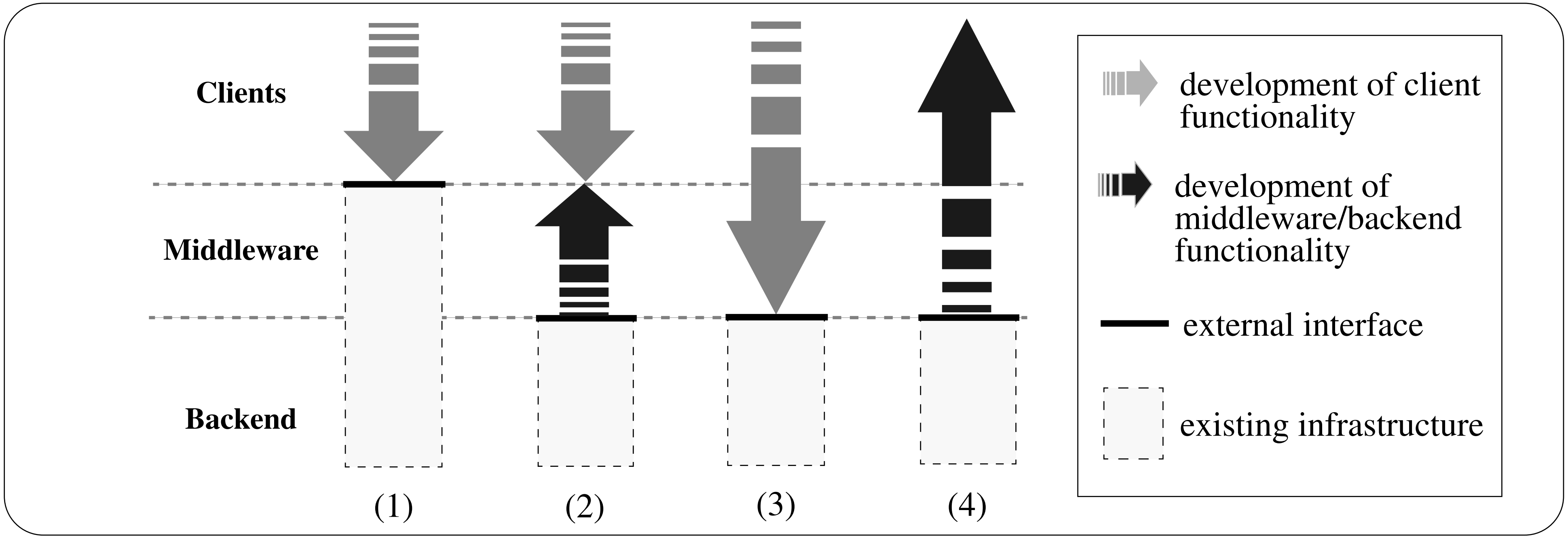
- Rule of thumb: The closer to the backend, the more burdensome the legacy.



- Mapping tools, CORBA, EJB, etc., provide means to abstract from legacy infrastructure, however only to a limited extent from legacy semantics.

- Migration paths to be determined on a case-by-case basis.

- New Applications and Legacy - common scenarios



Case	Functionality based on new	Characteristics
(1)	client application	common case; depending on the kind of middleware interface (e.g., CORBA services) the ideal view
(2)	client application and middleware interface	common case
(3)	client application	not very common; usually middleware required
(4)	backend functionality and terminal interface	still common (teller machines); probably prevailing

3.2 Application Requirements

- There are many factors that determine the requirements of applications (cf., literature on requirements engineering).

- The factors considered here are:

  - Longevity of applications,

  - Orientation of development process.

Longevity of Applications

- .... denotes the time span from development through production to “retirement”.

Longevity	Time Span
Short-term	< 2 years
Medium	2 to 5 years
Long-term	> 5 years

- The number of short-term to medium-term applications is increasing.

➨ Optimizing the process of application development is thus important.

Orientation of Development Process

- ... denotes whether an application is built from scratch or founded on existing infrastructure.

- On-Top Development:
  - Applications are built on top of existing, immutable infrastructure (legacy/ tradition), i.e., hardware, software, interfaces, or data.
  - Examples: Default case for application development.

- From-Scratch Development:
  - All data structures, functionalities, and constituents are built from scratch.
  - Examples: Very rare; almost too ideal an assumption.

- Unfortunately, software vendors often assume from-scratch development or view their products in an isolated fashion.

  ➡ Proper integration of new software technology often hard to achieve.

3.3 Current Technologies

- The following discussion again assumes three levels of processing.

Clients

- "Thin clients", Java

  ➡ Attempt to re-centralize processing.

- But how about, e.g.,
  - "Classical" client applications (e.g., word processing)?
  - Scalability of Java VMs, network connections, etc.?

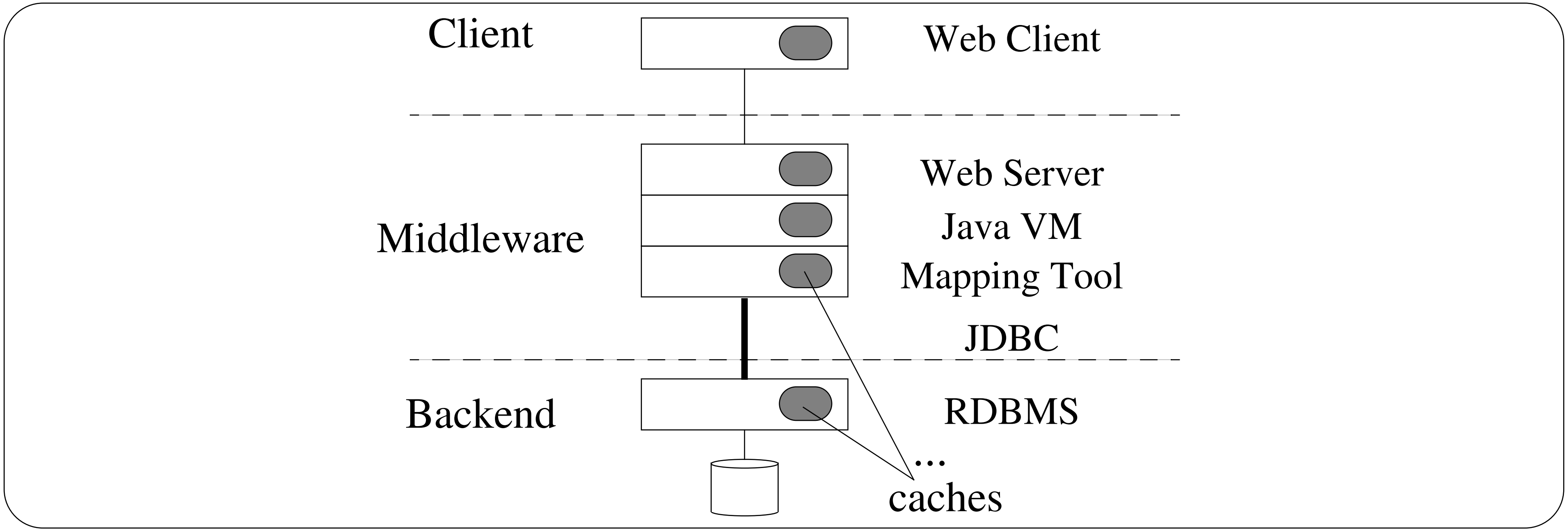
- Gartner Group: "The thin clients of tomorrow will be fatter than the fat clients of today".

- Most DB-related application processing located at the levels of middleware and backend.

Middleware

- Application servers and their services: CORBA, Java, EJB, Web, Documents, ...

- Sample application scenario



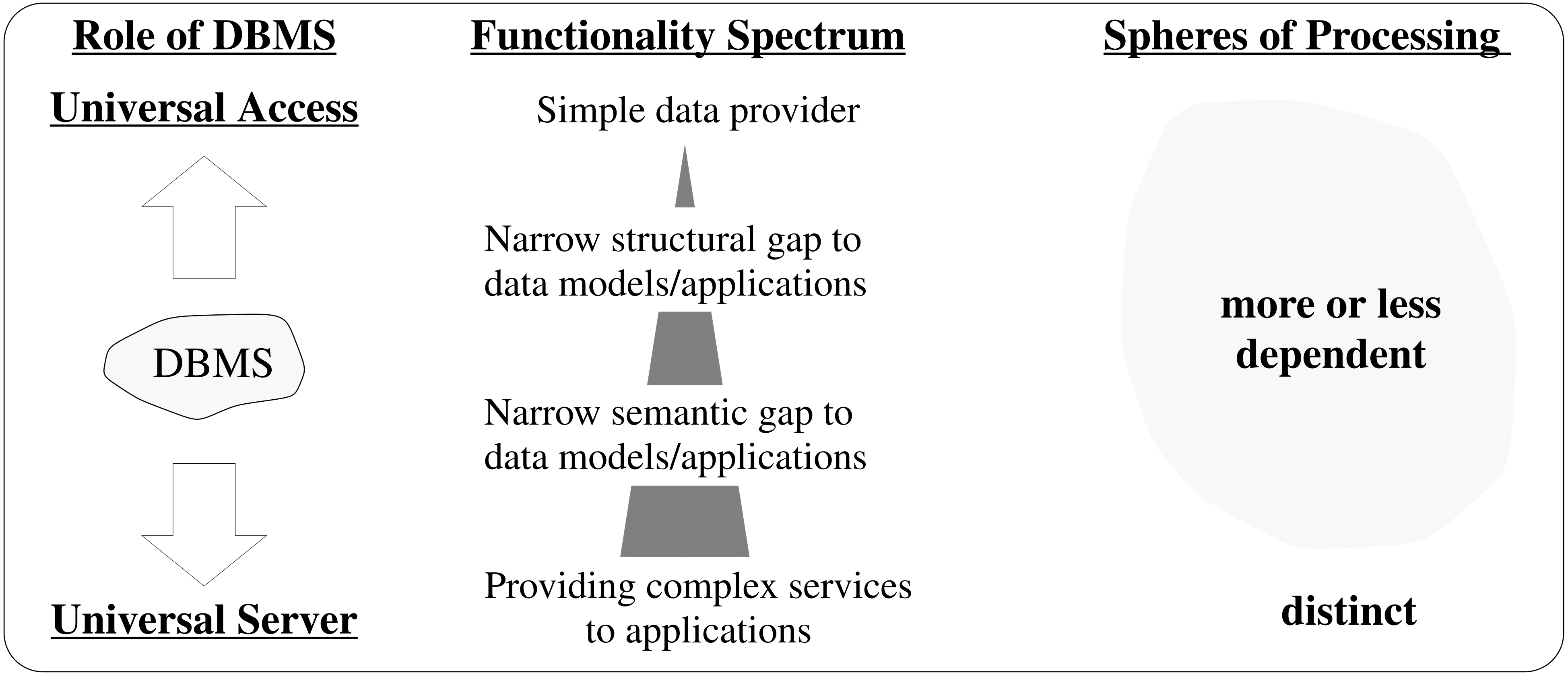
- Problems from a DB perspective
  - Transactional processing across multiple platforms/layers,
  - Repeated caching of DB data at several levels and in various formats.

➡ Coherency and optimization of processing across all spheres of control?

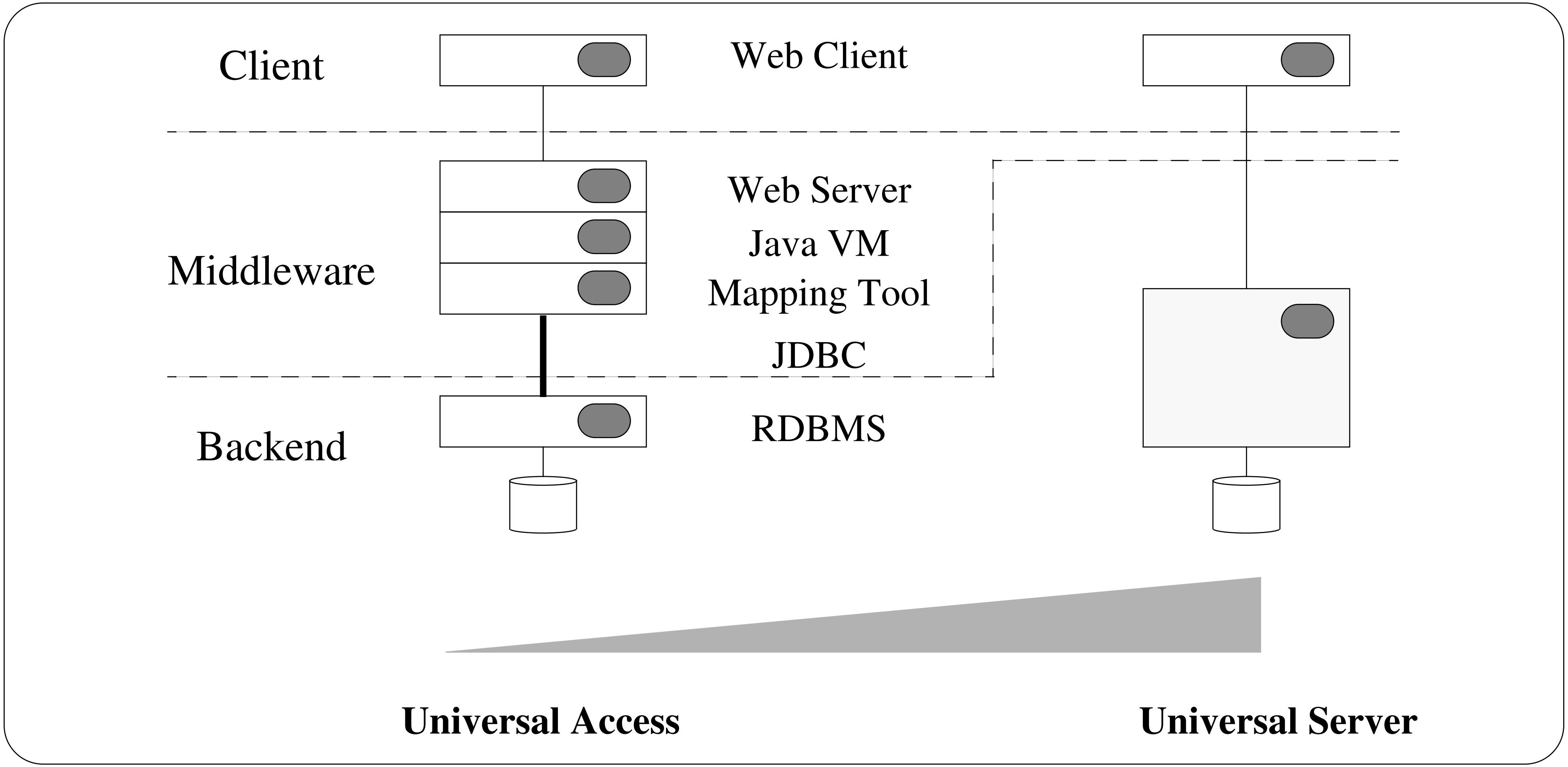
Backend

- Expressiveness of DBMS data models increases
  - SQL99, object-relational, extended relational

- Role of DB processingt?



- Possible scenarios:



4. Conclusions

- Historic evolvement gradually enriched concepts and technologies available for application modeling and processing.
  - ➨ Increased flexibility w.r.t. to technological issues and semantic aspects.
  - ➨ But also: Increased complexity.
- Key problem: How to keep applications technologically flexible, up-to-date, yet as simple as possible?
  - ➨ Role of DBMS needs to be clarified.
  - ➨ Coherency and optimization of processing across all spheres of control needs to be improved.