The Changing Software Paradigm

Gio Wiederhold

September 1995

What is changing

Programming is changing

from code generation

to composition of existing codes

Two views of the change:

1. **Logical progression**
2. **Paradigm change**

Effects of the change

- **New business models**
- **Need to adapt education**

1. Logical Progression

- Machine language 1945
  – generation of basic code sequences
- Assembly language 1955
  – shared code & shared data in memory
- Compiled languages 1960
  – standardized code units with parameters
- Database services 1975
  – control ceded to the DB administrator
- Object Libraries for specific Domains 1990
  – domains & solution structure predefined

Loss of autonomy & Cede control
Increase of scale & Benefit from work of others

Gio Wiederhold SWparadigm 3

1.a Invocation and reuse

data/code

- JUMP into, out of code
  – collect & reuse common code sequences
- TRANSFER EXECUTION control
  – subroutines with shared data in memory
- CALL statements
  – subroutines with parameters
- SQL to retrieve data from database services
  – shared historical data
- C++ CALLs with hidden domain structures
  – predefined domains & solution methods
- Information Services that mediate
  – Ex: *YaaHoo, Harvest, Price-Waterhouse K-10*

Gio Wiederhold SWparadigm 4

2. Paradigm Shift

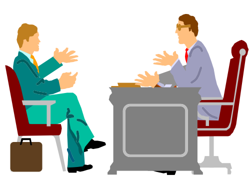
- Re-allocation of autonomy is gradual, *invisible*
- In the aggregate, over time, affects
  - Technology Paradigm
  - Business Practices
  - Educational needs
- Hypothesis:

We have to explicitly change from
a ***subroutine mentality*** (programmer control) to
a ***service mentality*** (cooperation & mediation)

Gio Wiederhold SWparadigm 5

Evidence

PAST ARGUMENTS	CURRENT ARGUMENTS
• Hardware architecture Number of bits/word Number of registers Size of address field • DEC vs IBM vs CDC	• Component architecture Description Asynchrony Inheritance • CORBA vs OLE vs ODE



ARGUING implies VALUE SHIFT from Hardware to Software Componentry

Gio Wiederhold SWparadigm 6

Early Example ~1980

Databases

- Provide a *Service* not under control of the application programmer
- Removed about 50% of code from commercial applications
- Changed training requirements of programmers
- Led to new sub-specialities:
  - » database administrators
  - » database programmers
  - » database educators
- Updates in the interface require negotiation

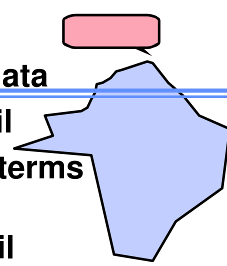
Database Interfaces

SQL:

two parts

1. a Non-procedural Query Language
2. a Schema which exposes available data

but hides much detail



The query language is restricted to the terms given in the schema

The schema hides implementation detail

- The Database Management System (DBMS)
  - – has the freedom to optimize query processing
  - – must optimize to achieve performance

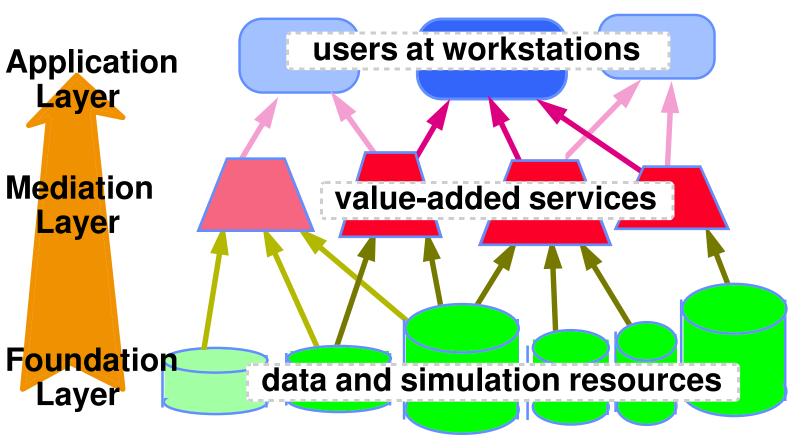
Recent Examples ~1990

Of removal of code by services:

- Object libraries { Microsoft, KPlib, GNU, NASA, ...}
  - » not autonomous in terms of control
- Graphical Interface (GUI) builders {MOTIF, . . . }
  - » procedural, no optimization
- Application builders {Powerbuilder, NeXTstep, Visual Basic}
  - » high-level programming based on standard interfaces
- Domain Specific Software Architectures (DSSA)
  - » structure, rather than code
- Mediation Architecture {middleware}
  - » intermediate, composable information services

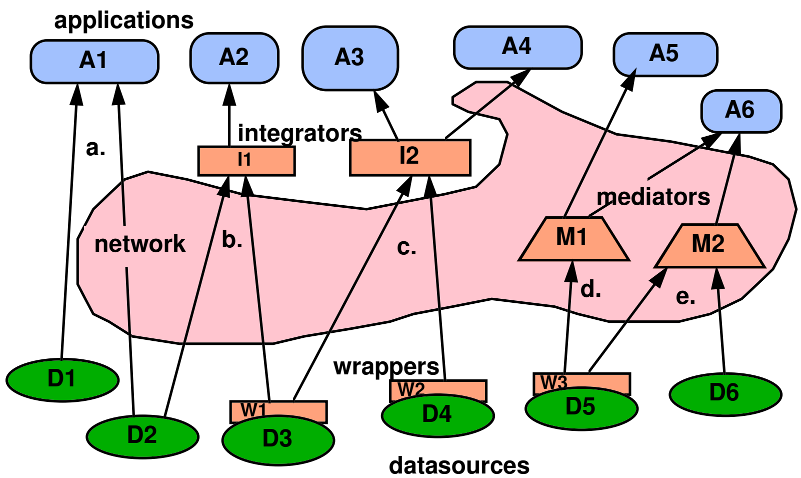
Gio Wiederhold SWparadigm 9

Transform Data to Information



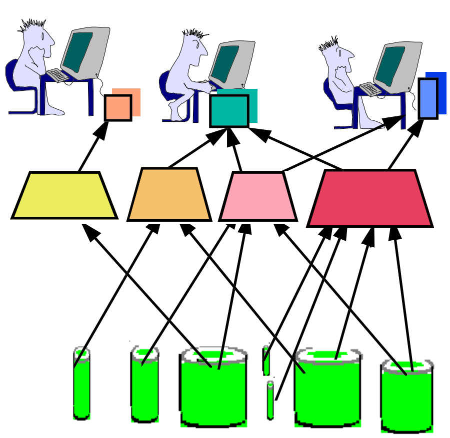
Gio Wiederhold SWparadigm 10

Evolution of mediation



Gio Wiederhold SWparadigm 11

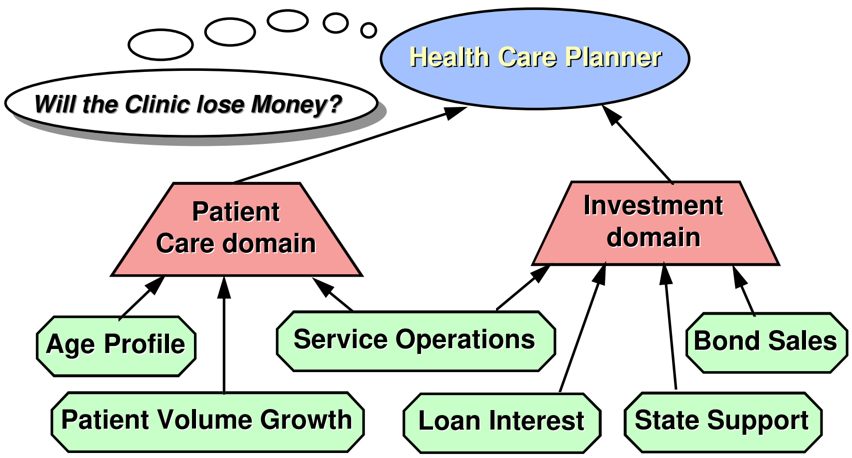
Domain-specific Mediation



- **User application**
  - Workstations
- **Mediator**
  - Expert-owned nodes
- **Data sources**
  - Remote primary and byproduct services

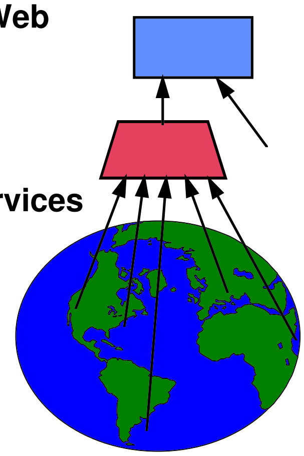
Gio Wiederhold SWparadigm 12

Example in Health Care



Gio Wiederhold SWparadigm 13

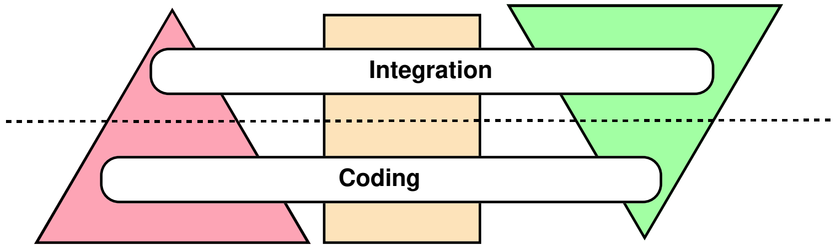
Mediation on the WWW



- Resources on the World-Wide-Web
  - are plentiful
  - autonomous
  - incoherent
- Opportunity for value-added services
  - select best source
  - improve coverage
  - minimize overlap
  - resolve inconsistencies
  - summarize results

Gio Wiederhold SWparadigm 14

Workforce reallocation



Integration originally performed for large systems (> 5Mbyte)

- by system integration companies:
  – Honeywell, IBM federal, Lockheed, SAIC, Andersen ...

Integration now performed for most systems (> 5Mbyte)

- by application clients and services:
  – *too many to name, including you . . .*

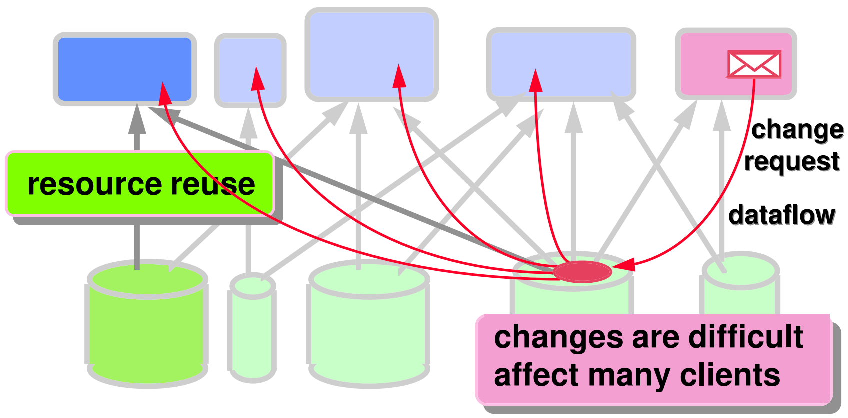
Dealing with Multiple Services

- For coverage, comparisons:
  – Multiple source databases within a domain
- For breadth
  – Databases from distinct domains
  (SAY: finance, productivity, product specs)
- For independence
  – Multiple suppliers
- Lack of coherence
  – Multiple standards (supplier vs customer interests)

Problem: must maintain flexibility

Growing Systems: *n* modules

Client-server: can have many servers and clients



Gio Wiederhold SWparadigm 17

Examples

- LL Bean Catalog
  - handled 3 credit cards: Visa, MC, AmEx
  - could not adapt quickly to Discover
    - » contributed to loss of business

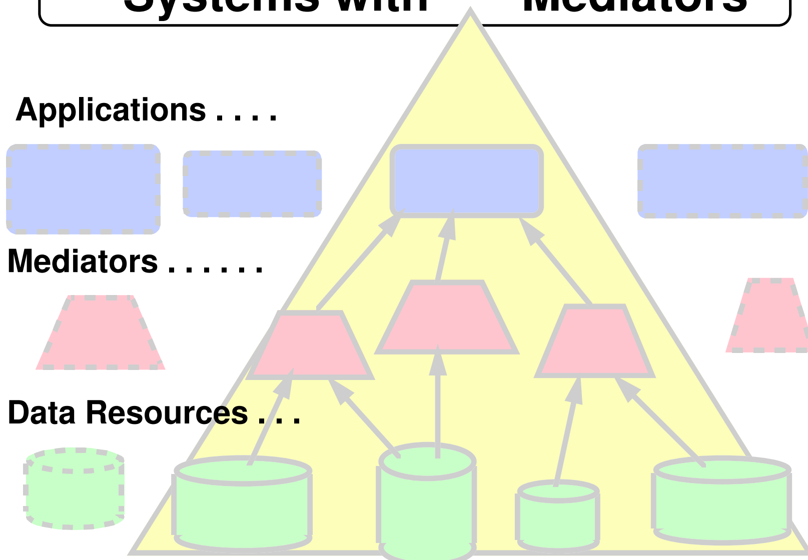
- WMCCS (Military inventory system)
  - 100’s of databases, multiple domains
  - 100’s of applications

  successful wrt to specifications

  but cannot be upgraded at all anymore

Gio Wiederhold SWparadigm 18

Systems with Mediators



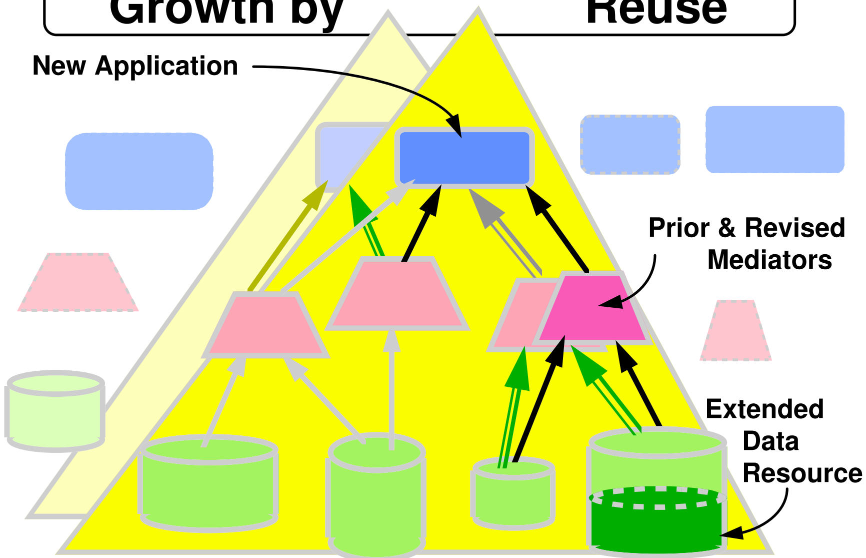
Gio Wiederhold SWparadigm 19

Emerging Mediation

- **Tools**
  - **Open server (Sybase)**
  - **Blades (Illustra)**
  - **Distributed Object Environment (Sun/Persistence)**
- **Initial Application Areas**
  - **Intelligence systems**
  - **Flexible manufacturing**
- ***Future***
  - ***SimQL to retrieve projections into the future***
    - » ***information provided by experts & simulations***

Gio Wiederhold SWparadigm 20

Growth by Reuse



Gio Wiederhold SWparadigm 21

Incremental Maintenance

1. New/Revised application neeeds more data
2. Mediator isolates other applications
3. New application checks out data and function
4. Other applications can join when feasible

versus

1. New/Revised application neeeds more data
2. Determine affected applications
3. Change affected codes, create test scaffolding
4. Plan for system-wide release at *half-year* cycle
5. Batch all other changes for that cycle
6. Perform critical, risky release of batch

Gio Wiederhold SWparadigm 22

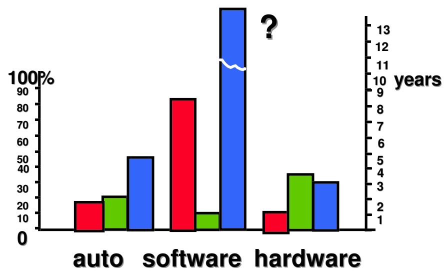
Maintenance Metrics

Want good BENEFIT/COST

- **Benefits** to customer
  - usage if choice
  - *happiness?* if no choice

Depend on relevant information, requires constant updating
  - leads to growth
  - growth leads to heterogeneity

- **Cost**
  - initial
  - maintenance (60-80% SW)



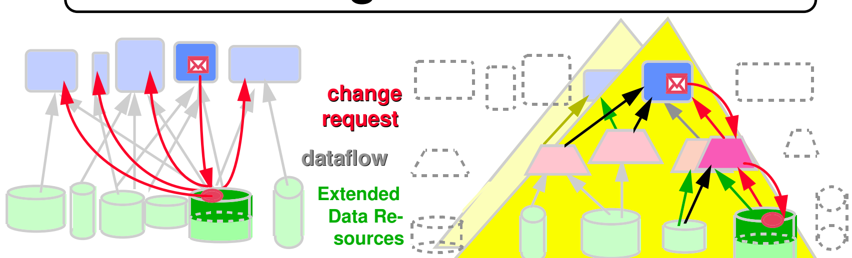
long lifetime

low depreciation = 1 / lifetime

high maintenance cost

Gio Wiederhold SWparadigm 23

Measuring architectures



Client-Server:
- Direct resource reuse
- Changes affect many clients
- Cost is $\mathcal{O}(m^2)$?
- Updates are batched

Mediated:
- Indirect resource reuse
- Mediator knowledge reuse
- Changes in 1 mediator
- Cost is $\mathcal{O}(3m)$?
- Updates are incremental

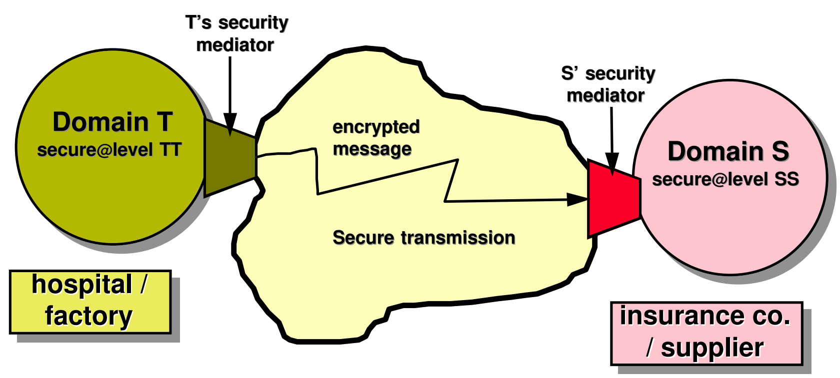
Gio Wiederhold SWparadigm 24

New Metrics

- Productivity
  Lines of Code --> Time to Asssemble a Product
- Time to Update Product
  Several years --> 3/6 months
- Satisfaction of
  System Administrator --> *Customer*
  – -- *missing:*
- Performance based on
  Detailed data --> Component / Service specs

Gio Wiederhold SWparadigm 25

Interoperating Domains



Transparency
- to network conventions
- to databases within domains

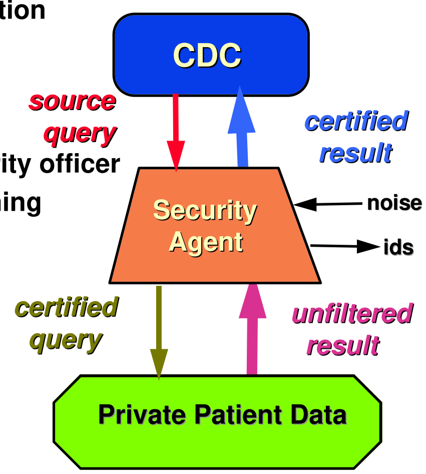
Gio Wiederhold SWparadigm 26

Mediator for the Security Officer

Example: Public Health Application
- Valid statistical data
- No access to private data

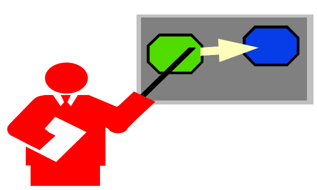
Security Mediator
- Owned by independent security officer
- Does aggregation and screening
- Operates be default Manually
- Enables gradual automation

Physicians' Databases
- Valuable resources
- Need to be aggregated for significance

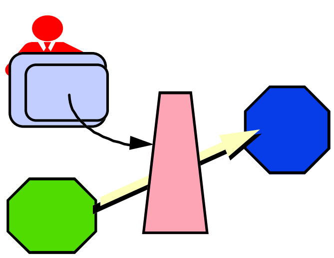


Gio Wiederhold SWparadigm 27

New Role for Consultants



Old
- Used at Design Time
  and
- To Explain Failures



Future
- Available as a Service
- Responsible for Knowledge Maintenance

Gio Wiederhold SWparadigm 28

Conclusion

- **Change in software paradigm to *Composition* is occurring, nearly invisibly**
- **Flexibility is becoming more important than satisfying all wishes wrt to performance and features**
- **Mediation provides incremental, responsive, and less risky maintenance at less cost**
- **Old habits and metrics of success are becoming obsolete and a hindrance**

Acceptance of Service Paradigm