

# A Visualisation System for a Peer-to-Peer Information Space



TAMPERE UNIVERSITY OF TECHNOLOGY

*Hypermedia Laboratory*

Ossi Nykänen, Jaakko Salonen, Matti Haapaniemi and Jukka Huhtamäki  
Tampere University of Technology, Hypermedia Laboratory

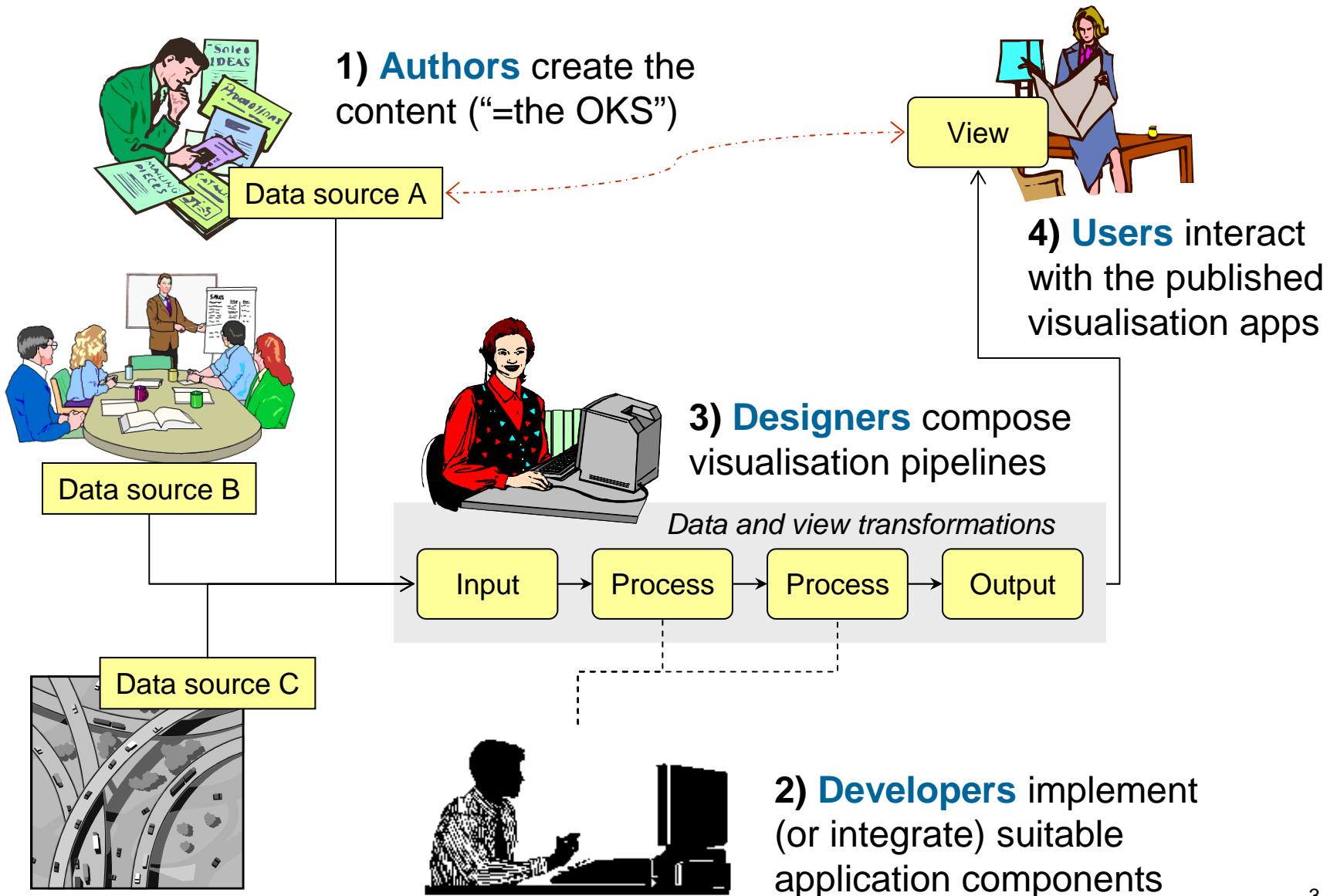
**Presentation at the  
OPAALS 2008 Conference  
7-8 October, Tampere, Finland**



# Introduction

- The objective of this work is to study (and implement the core components of a) **lightweight framework for implementing data-driven visualisations**
- Two main phases of work (part of the OPAALS NoE)
  - Past: Phase 1 & proof-of-concept prototype
  - Currently: Phase 2 & Peer-to-peer component-based visualisation system
  - (Phase 3: continued R&D)
- Presentation outline
  - Pipeline-based visualisations
  - Peer-to-peer (P2P) visualisations
  - Searching in P2P
  - Design rationale for Phase 2 visualisations
  - Discussion

# Basic idea of data-driven pipeline visualisations



# Peer-to-Peer networks?

- A **Peer-to-Peer (P2P) system** is a self-organising system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services (Steinmetz and Wehrle, 2006)
  - Intuitively this means a (technical) network infrastructure without (too) centralised components
- **Abstract tasks** for P2P systems include storing/retrieving (in a distributed, robust manner)
  - data
  - metadata (keywords or semantic descriptions)
  - services
- **Physical placement** (of e.g. data) is typically done according to *ownership* or *search strategy*
  - Note that metadata may include references to resources external the P2P system (!)

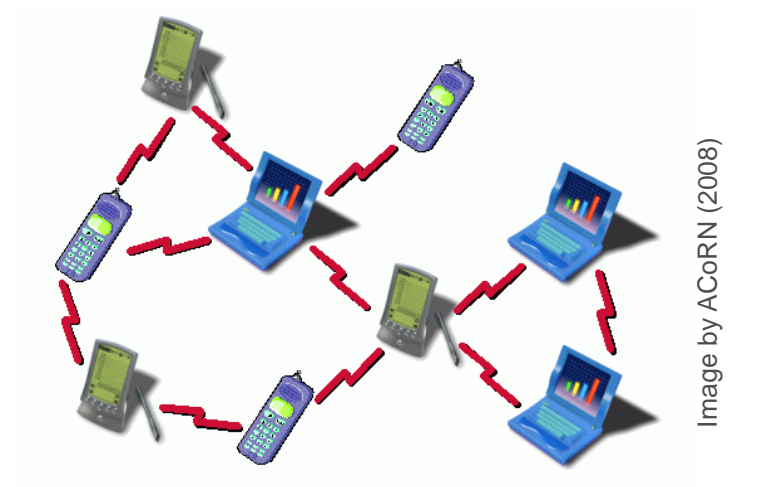
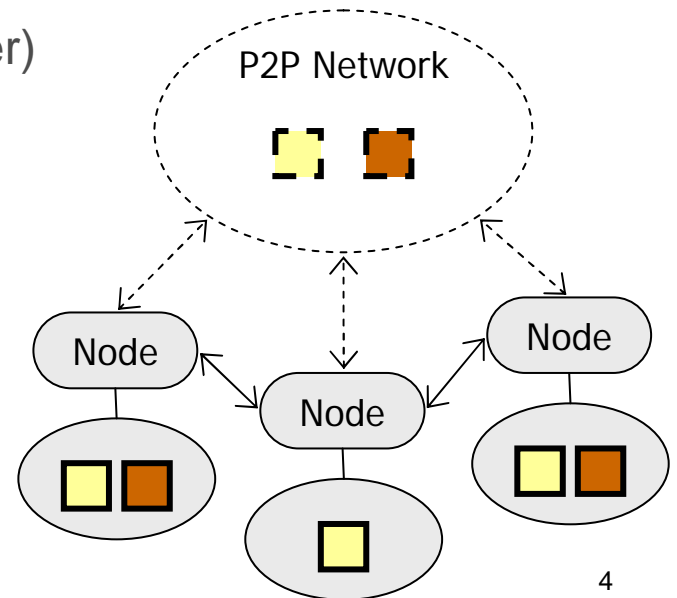


Image by ACoRN (2008)



# Peer-to-peer visualisations?

- Levels of P2P distribution

1. Published visualisations (e.g. social network application)
2. Pipeline definitions (e.g. Ant task)
3. Source data (e.g. information about participants)
4. Components of a pipeline (e.g. graph transformer)
5. Pipeline processor itself (thus migrating into, e.g., constraint-based service orchestration)

- In an Internet context, the Phase 1 visualisation prototype already manages the levels 1-3

- In order to reach the level 4, Phase 2 visualisation application treats pipeline steps as **services**

- Note that a pipeline definition may be perceived as source data, and source data can in turn be treated as a kind of service

- Level 5 distribution is not pursued since the main use cases can be reasonably captured within levels 1-4 (and pipeline processing may itself be published as a service). Implications:

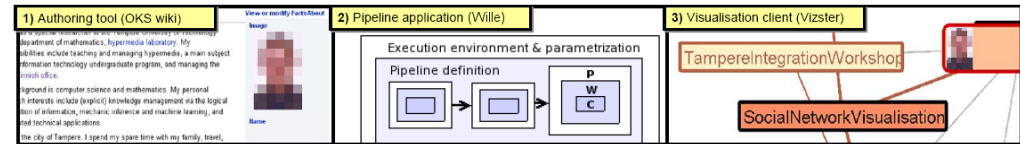
- A logical pipeline is always computed within a single node
- (Implicitly) hierarchical pipelines are possible (since steps may be (sub)pipelines)

- Resulting **requirements** to the P2P architecture

- Common data model, access to data and services, and **semantic search**

- Why semantic search? Because in general, four levels of abstractions may be identified when requesting a service component (including “source data”):

1. Directly by **locator** (node identifier, service identifier).
2. Indirectly by service **identifier**. (When several nodes provide identical services.)
3. Indirectly by service **category**. (General-purpose services of different implementations.)
4. By service **description**. (The most general case that nevertheless requires sufficiently detailed descriptions of services and some a priori agreements about their semantics.)



# Searching in P2P (“for the practically-oriented”)

- In the most simple form, a **key-based** P2P system simply performs **lookup**, i.e., identifier allocation and I/O:

```
P2PNode myNode = P2PNode(network,username, password)
```

```
Document mydoc = Document(“Hello World!”)
```

```
P2PIdentifier myid = myNode.push(mydoc)
```

```
...
```

```
Resource doc = myNode.fetchByKey(myid)
```

- The problem is obvious: in a lookup system, one needs to know the P2P identifier *a priori* (!)
- (With respect to data retrieval, but not, e.g., lookup efficiency) the “next better” P2P system is a **keyword-based** one:

```
...
```

```
List resourceList = myNode.fetchByKeyword(“biology”)
```

- Finally, the “best” system supports (also) **schema-based** searches:

```
...
```

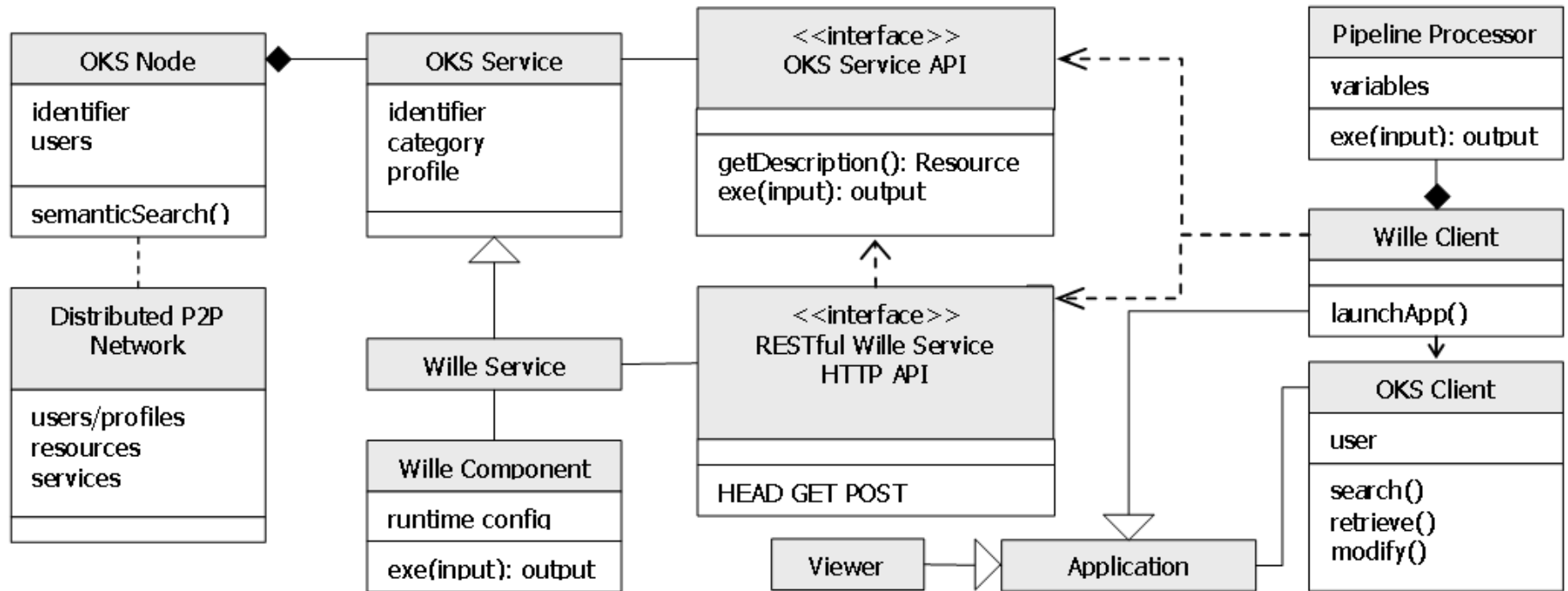
```
List resourceList =
```

```
myNode.fetchBySchemaPattern(“dc:subject=‘biology’ dc:author=‘Lamarck’”)
```

# Design rationale for Phase 2 P2P visualisation system

1. Visualisation components (=services) should be useful as such, without always having to install a specific pipeline processor client to exploit them. (This also supports collaboration between non-technicians.)
2. There should be a way to easily create new visualisation components from the existing data processing applications, and offer these as visualisation service components using an appropriate service platform. (=service wrappers)
3. Besides P2P, the API of visualisation components should follow the conventions of REST when applicable. Integrate components without always having to implement complex adapter or wrapper systems. (=integrate with existing Web applications)
4. Visualisation components are not (in general) responsible for storing (intermediate/final) results or their computations – this is the responsibility of the visualisation client. (=integrate with existing processors)
5. Visualisation clients are defined by their ability to meaningfully exploit the services of visualisation components, not by their internal making. (Common framework is helpful, though.) (=pipeline-centric thinking is sometimes a burden, think about visualisation clients instead)
6. Visualisation clients (acting as pipeline processors) typically fall into three main categories: command-line scripting, local application, and web application (=from the perspective of the user interface, there is little new)

# Architecture for P2P visualisation services



# Discussion & conclusion

- During our research, we have reasonably well specified a P2P visualisation system and identified the next steps
- Experiments with the current P2P systems (Gnutella, Chord, Sirona) reveal that the two key features are problematic or missing (“from the OKS”):
  1. service deployment
  2. semantic search
- As a consequence – and in order to claim that we are developing a P2P visualisation system in a first place – a (simple HTTP/RMI-based) proof-of-concept visualisation service framework was developed as well
  - This ideology nicely integrates to P2P with a semantic search (“publish service descriptions and service locators if nothing else”)
  - However, the underlying OKS infrastructure **should** include the functionality 1&2 above – this is needed for applications in general, not just visualisation !
- The next steps of our work include developing visualisation client framework (incl. few core components) and a simple end-user client
- The work then continues by reporting selected illustrative scenarios (including a conference visualisation scenario) and deploying the final system
- Thank you! Questions/comments?