Softwaretechnologie für Fortgeschrittene
Wohce 1
Algorithmen
Modellierung

(with contributions from Christian-Emil Ore, Jon Holmen, and other colleagues at the Unit for Digital Documentation, University of Oslo
and from Martin Dörr and Stephen Stead, CIDOC-CRM SIG)

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

Theckenperger

Tegenberger

Tekekenperger

Soundex

1 Das erste Zeichen jedes Namens wird beibehalten.

2 W und H werden ignoriert.

3 A, E, I, O, U und Y ergeben keinen Codewert, gelten jedoch als "Trenner" (s.Regel 5).

4 Die anderen Zeichen werden nach folgenden Regeln umgewandelt.

4.1	B, P, F, V	==>	1
4.2	C, G, J, K, Q, S, X, Z	==>	2
4.3	D, T	==>	3
4.4	L	==>	4
4.5	M, N	==>	5
4.6	R	==>	6

5 Ergeben zwei aufeinanderfolgende Zeichen denselben Code, wird er nur einmal gewertet. Sind sie durch einen "Trenner" (s. oben Regel 3) getrennt, wird er jedoch wiederholt.

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

Theckenperger

Tegenberger

Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

Theckenperger ➔ T251

Tegenberger ➔ T251

Tekekenperger ➔ T225

Algorithmen: Definition

Ein Algorithmus ist eine Funktion $f(D_{ein}, D_{aus})$, die *Eingabedaten* D_{ein} in *Ausgabedaten* D_{aus} schrittweise transformiert und dabei bestimmte Bedingungen erfüllt.

Algorithmen: Eigenschaften

1. **Exaktheit.** Die Funktion f kann präzise auf formale Weise beschrieben werden.
2. **Finitheit.** Die Beschreibung von f ist endlich lang.
3. **Vollständigkeit.** Die Beschreibung von f umfasst alle vorkommenden Fälle.
4. **Effektivität.** Die Einzelschritte sind elementar und real ausführbar.
5. **Terminierung.** Die Funktion f hält nach endlich vielen Schritten an und liefert ein Resultat.
6. **Determinismus.** Die Funktion f liefert bei gleichen Eingabewerten stets das gleiche Ergebnis, wobei die Folge der Einzelschritte für jeden Eingabewert genau festgelegt ist.

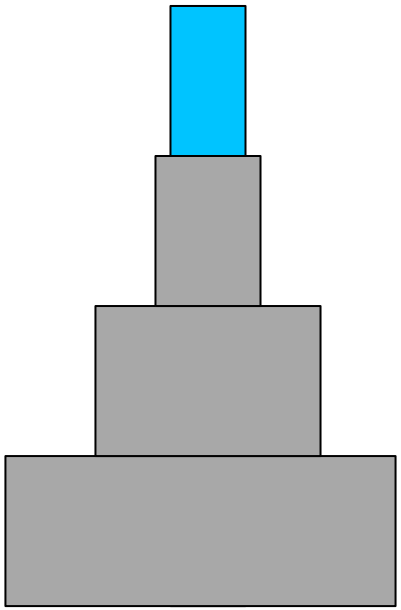
Towers of Hanoi

Situation in einem Tempel in Hanoi:
- Ein Turm von 100 Scheiben auf einer Spindel (S1).
- Eine leere Spindel (S2).
- Eine weitere leere Spindel (S3).

Transportiere S1 so nach S2 - wobei S3 als Zwischenlager verwendet werden darf - dass:
- Jeweils nur die oberste Scheibe von einem Turm genommen wird.
- Niemals eine größere Scheibe auf einer kleineren liegt.

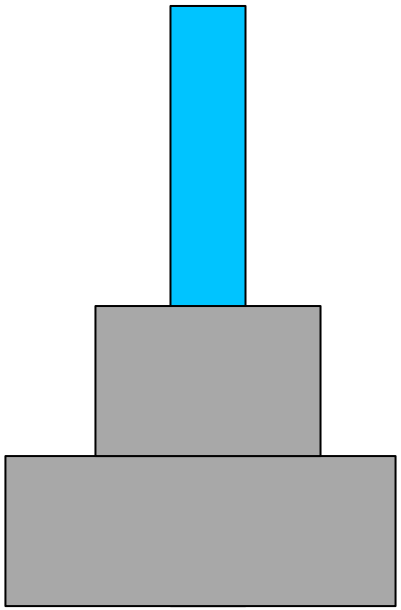
Prophezeiung: Ist das erledigt, ist das Ende der Welt gekommen.

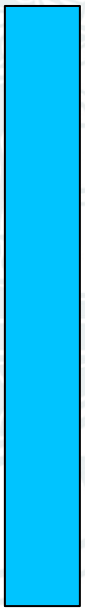
Towers of Hanoi



S1 S2 S3

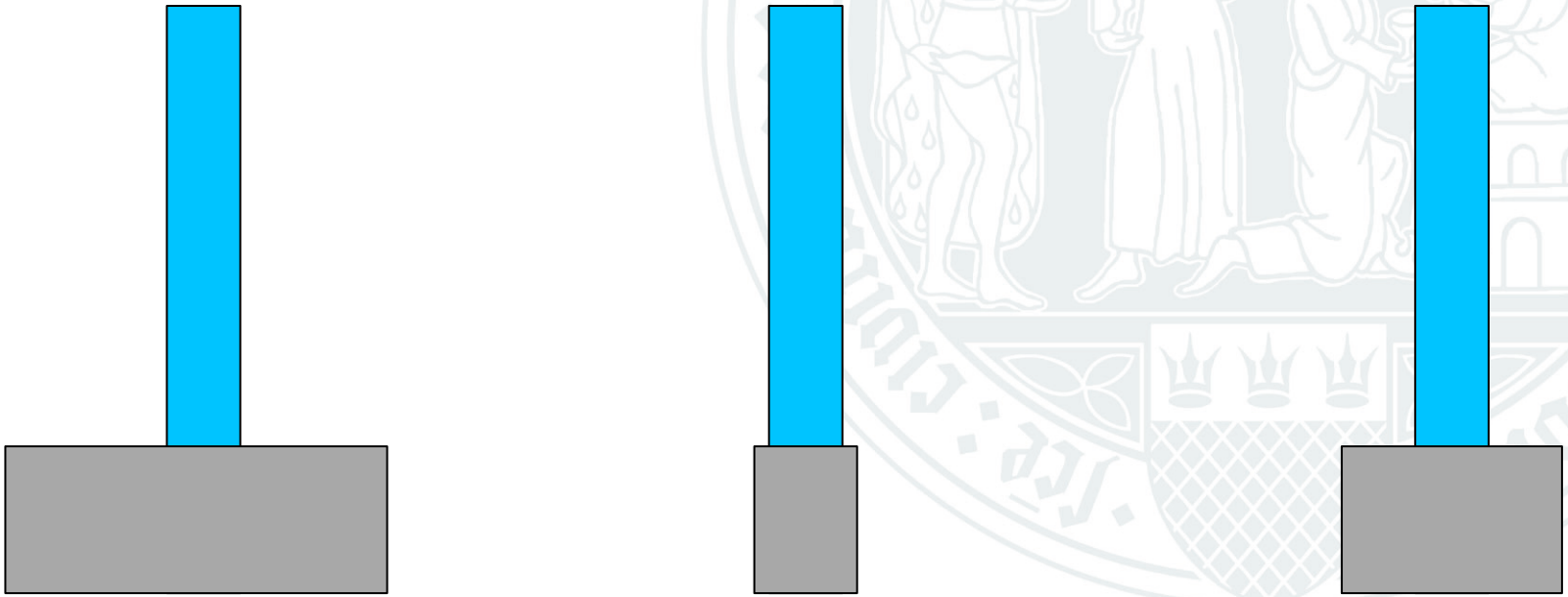
Towers of Hanoi





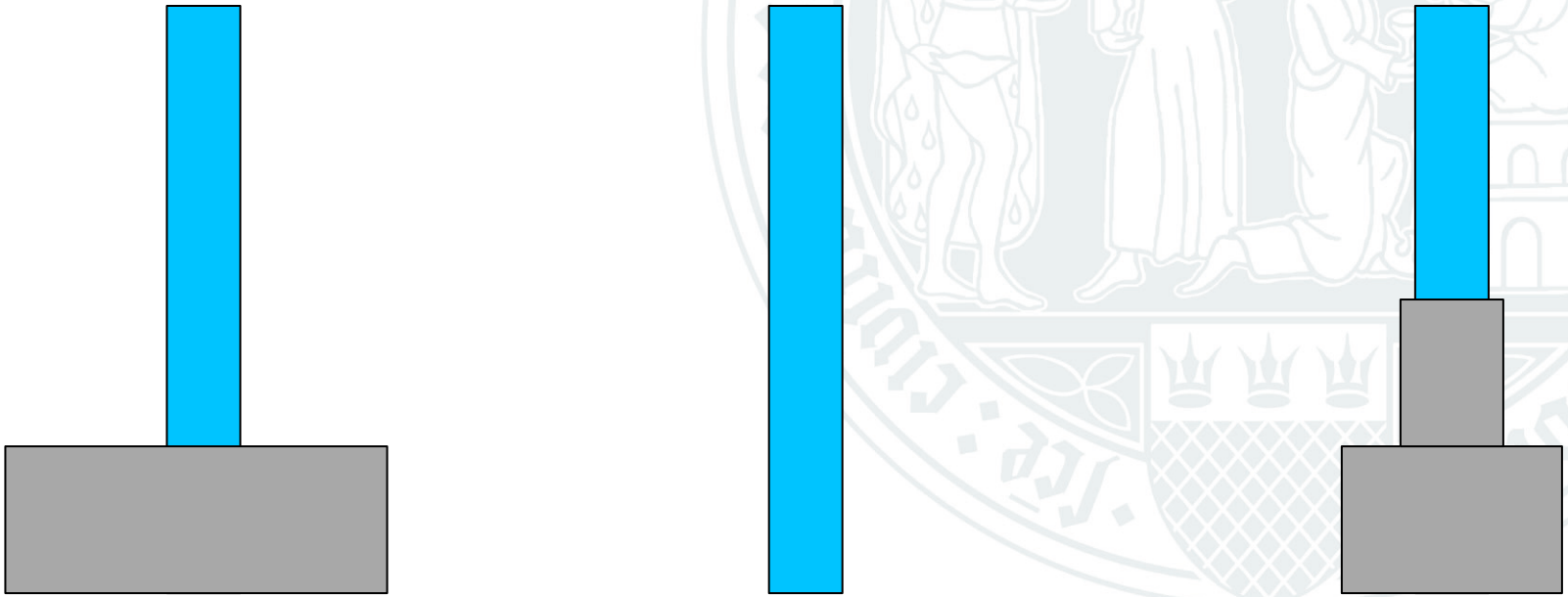
S1 S2 S3

Towers of Hanoi



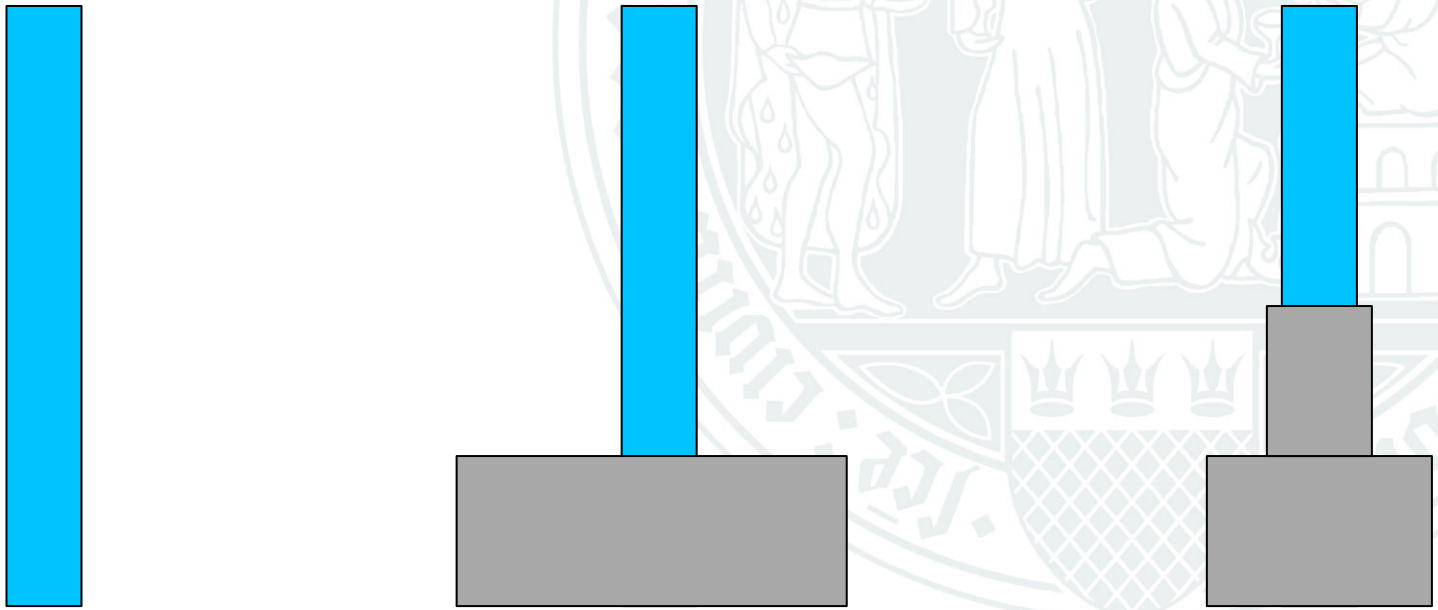
S1 S2 S3

Towers of Hanoi



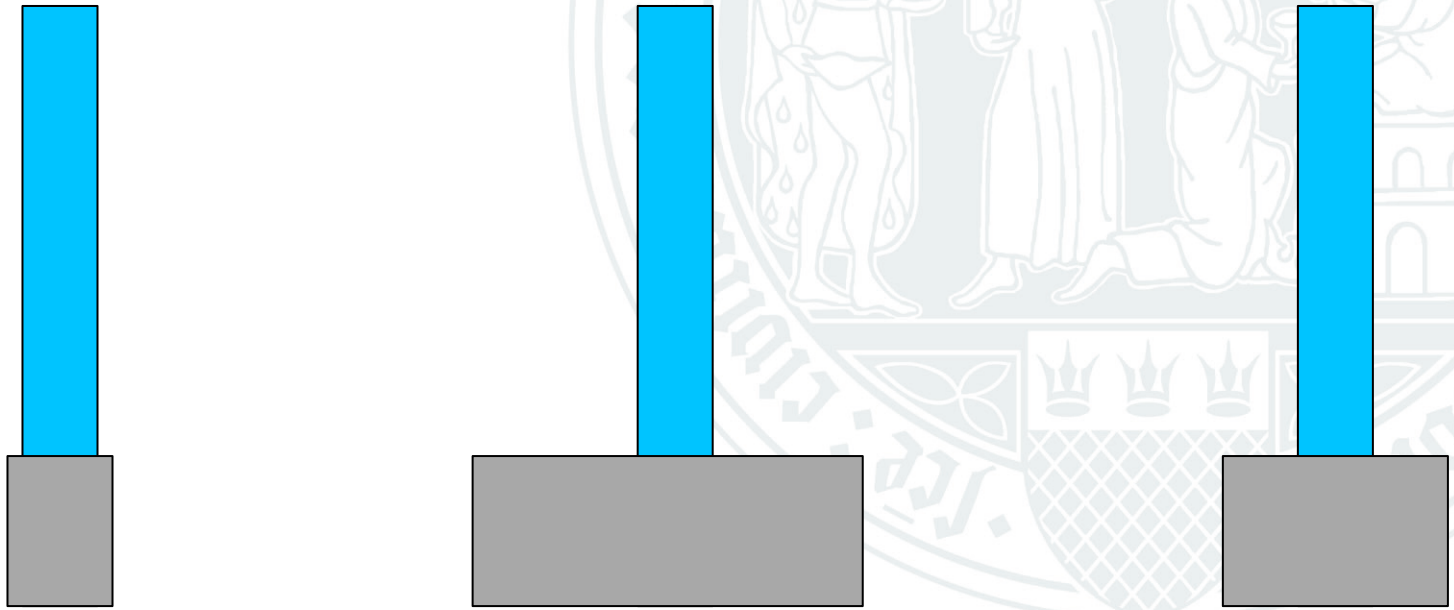
S1 S2 S3

Towers of Hanoi



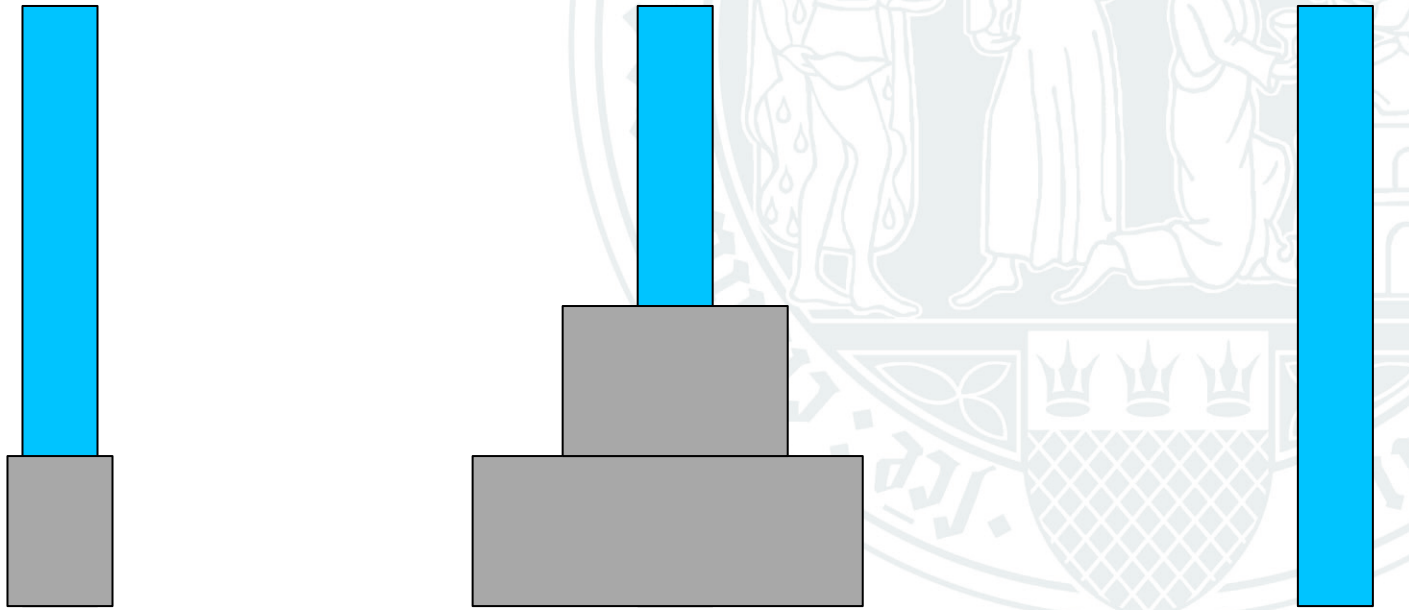
S1 S2 S3

Towers of Hanoi



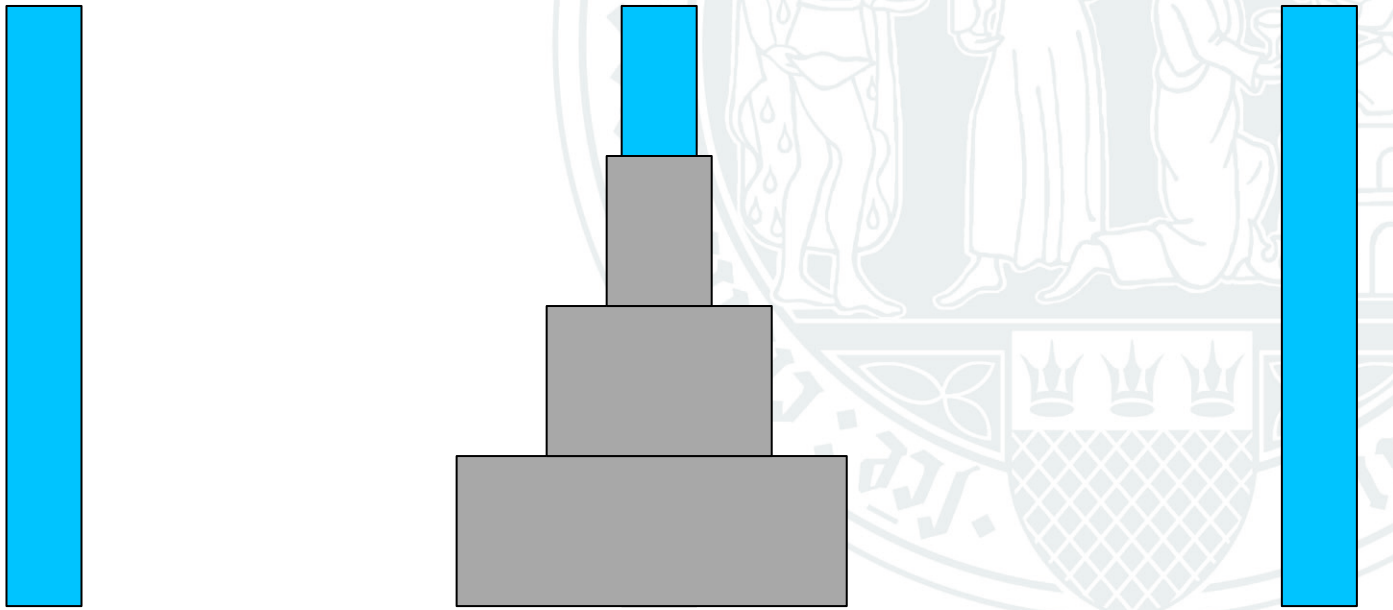
S1 S2 S3

Towers of Hanoi



S1 S2 S3

Towers of Hanoi



S1 S2 S3

Towers of Hanoi

Lösung I

1. Finde jemand, der die obersten 99 Scheiben von S1 nach S3 transportiert.
2. Transportiere die unterste Scheibe von S1 nach S2.
3. Finde jemand, der die obersten 99 Scheiben von S3 nach S2 transportiert.

Towers of Hanoi

Lösung II

1. Besteht der zu transportierende Turm aus mehr als einer Scheibe, finde jemand, der einen Turm von n-1 Scheiben von S1 nach S3 transportiert. Nutze S2 als Zwischenablage.
2. Transportiere selbst die unterste Scheibe von S1 nach S2.
3. Besteht der zu transportierende Turm aus mehr als einer Scheibe, finde jemand, der einen Turm von n-1 Scheiben von S3 nach S2 transportiert. Nutze S1 als Zwischenablage.

Towers of Hanoi

Lösung III

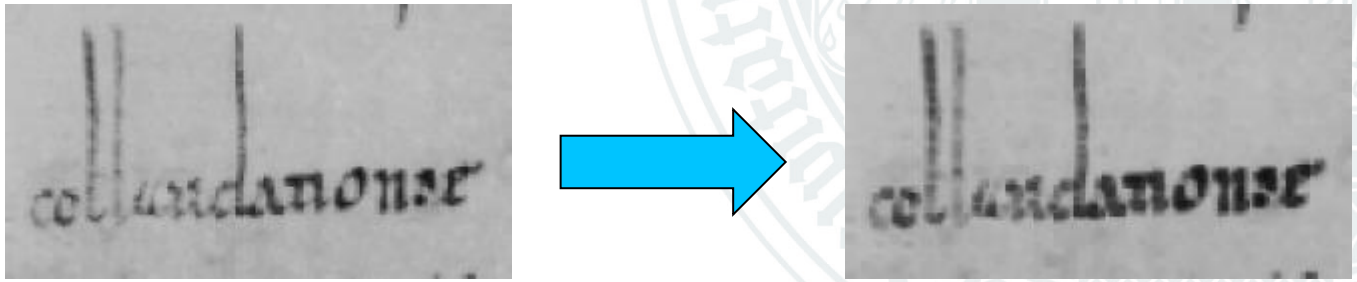
```
function transport( int n, stack spindel1, stack spindel2, stack spindel3) {
    if (n >1) transport(n-1,spindel1,spindel3,spindel2);
    schritt(spindel1,spindel2);
    if (n>1) transport(n-1,spindel3,spindel2,spindel1);
   }

function schritt( stack spindel1, stack spindel2) {
   spindel2.push(spindel1.pop());
  }
```

Towers of Hanoi

Fragen

1. Wie viele Mitarbeiter werden benötigt?
   n
2. Wieviele Transferschritte?
   2^n -1
3. Wie lange?
   2^{100}-1 Schritte == ca. 10^{30}
   1 Schritt == 1 Sekunde ==> ca. 10^{30} Sekunden == ca. 4 * 10^{22} Jahre

Minimal neighbour



Original Ergebnis

Minimal neighbour

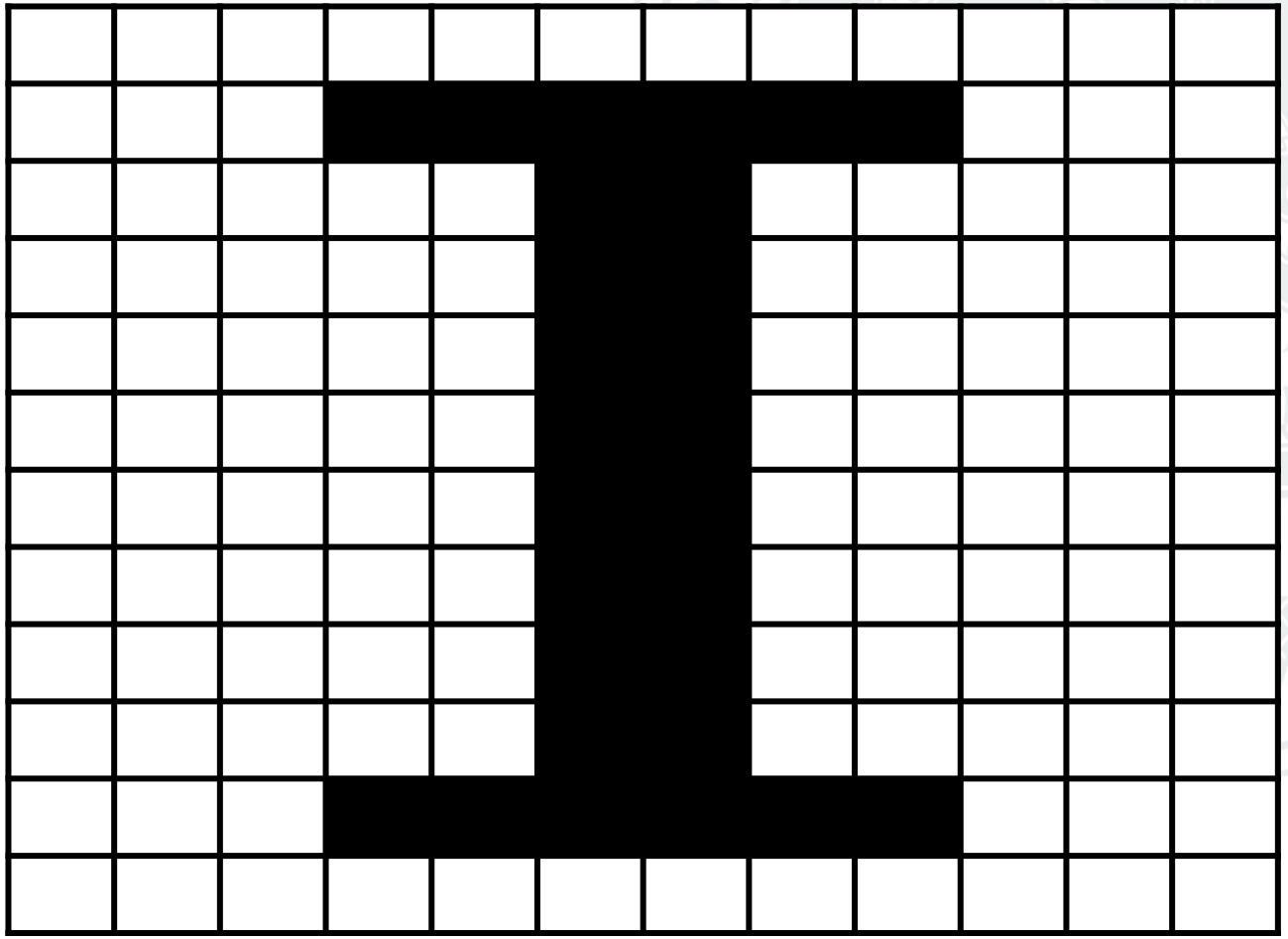
Ersetze in jeder Zeile

jedes Pixel

durch den niedrigsten Pixelwert der dieses

Pixels umschreibenden 3 x 3 Matrix.

Minimal neighbour



Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	50	50	50	50	50	50	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	50	50	50	50	50	50	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	250	250	250	
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	50	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

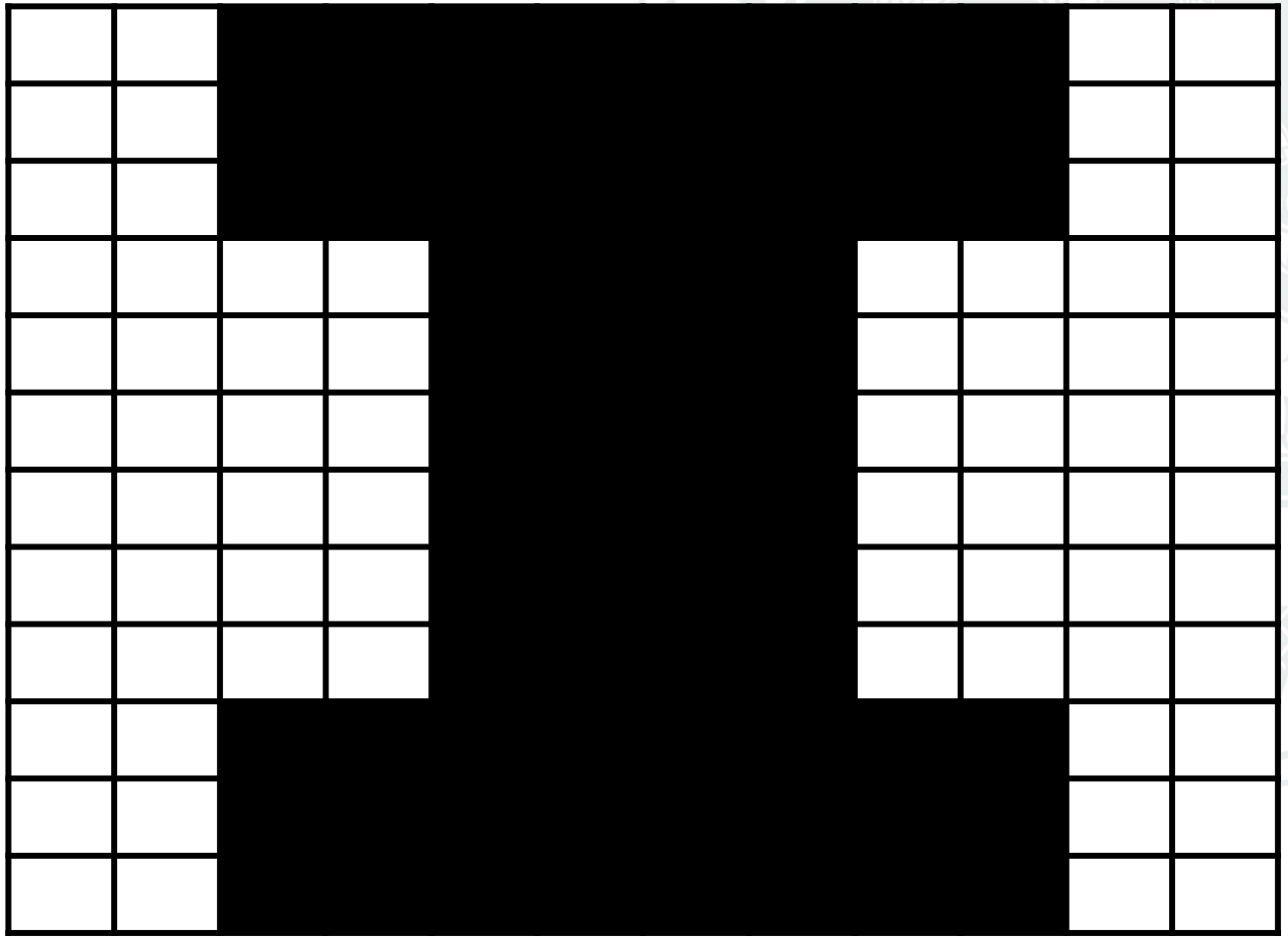
Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	50	**50**	**50**	**50**	**50**	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	***50***	***50***	***50***	***50***	***50***	***50***	***50***	***50***	250	250
250	250	***50***	**50**	**50**	**50**	**50**	**50**	**50**	***50***	250	250
250	250	***50***	***50***	***50***	**50**	**50**	***50***	***50***	***50***	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	250	250	***50***	**50**	**50**	***50***	250	250	250	250
250	250	***50***	***50***	***50***	**50**	**50**	***50***	***50***	***50***	250	250
250	250	***50***	**50**	**50**	**50**	**50**	**50**	**50**	***50***	250	250
250	250	***50***	***50***	***50***	***50***	***50***	***50***	***50***	***50***	250	250

Minimal neighbour



The media server

What are the requirements for a system for media objects?

- Upload
- Storage
- Metadata
- Presentation
- Long term preservation

Will focus on images but equally relevant for other media types

Upload operations

- Connect to the storage
- Find a logical place to put the data
- Submit metadata
- Establish a stream connection
- Upload bits
- Check result
- Get a receipt and an identifier back
- *The client may be:*
  - *a human*
  - *a computer programme*

Storage operations

- Receive a request with metadata
- Return an ID for the stream
- Receive the stream
- Receive further metadata
- Enter metadata into database
- Store file based on stream on disk
- Establish link from database to disk file

Metadata

- Technical metadata such as
  - file format
  - shoot date and time
  - size
  - location and direction
- Content metadata such as
  - motive
  - classification
  - date and time
  - source
  - location

Presentation

- User interfaces for
  - searching
  - listing metadata and/or thumbnails
  - delivering images in different formats
  - protecting images
  - ordering images
  - payment
- Different platforms
- Different user groups
- Different contexts

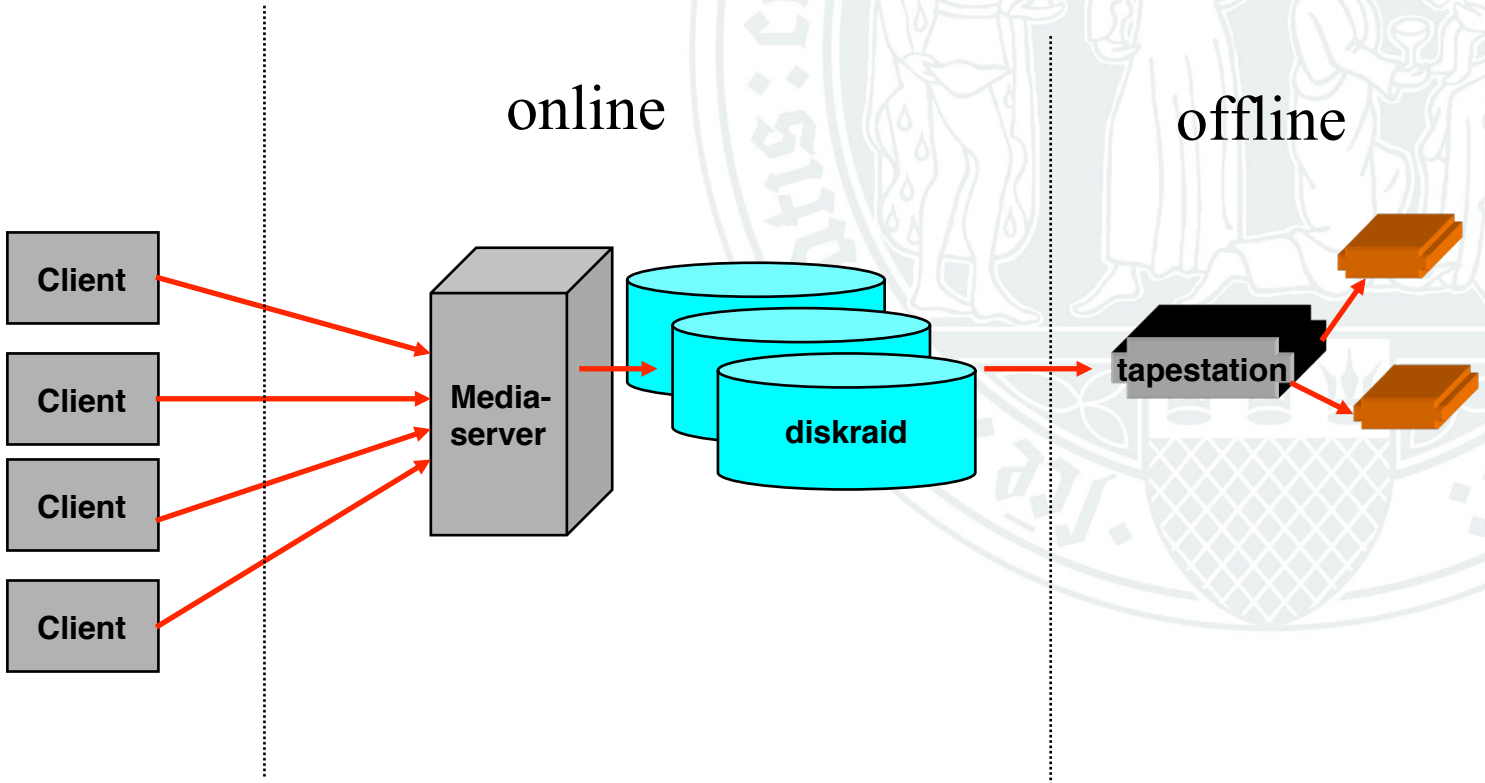
Long term preservation

- Make sure the data survives for the future
- *Long* term – not just 10 or 30 years
- Preservation
  - bitstreams
  - meaning
  - context
  - usability
- Technology
- Administration
- Politics

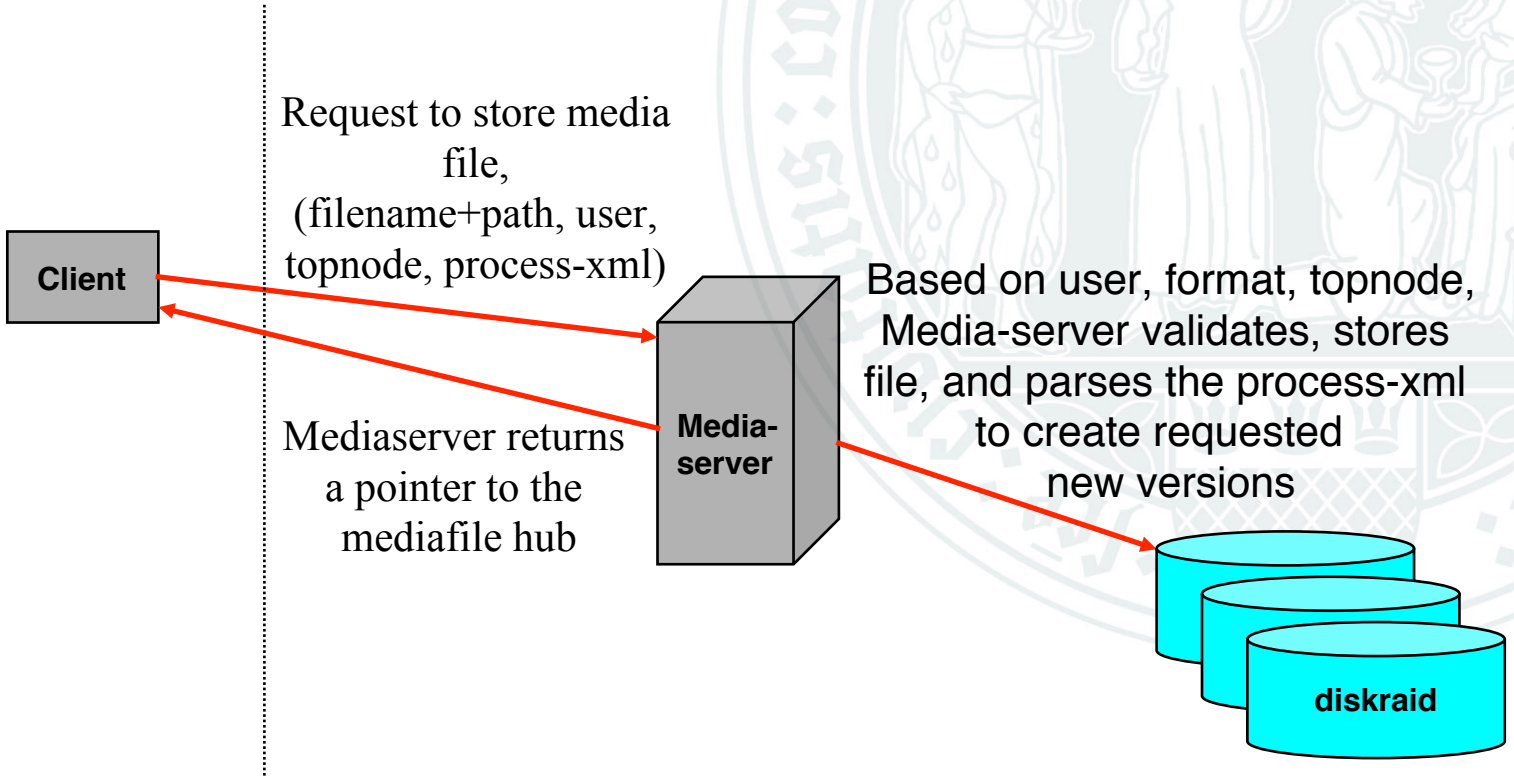
The system presented here

- Image collections at Norwegian universities
  - University history
  - Art history
  - Cultural history
  - Archaeology
  - Natural history
  - …
- Document archives
  - archaeology
  - dialectology
  - …
- Sum:
  - 1-2 000 000 traditional digitised photos
  - 3 000 000 document facsimiles
  - *(figures a few years old)*

Overall architecture



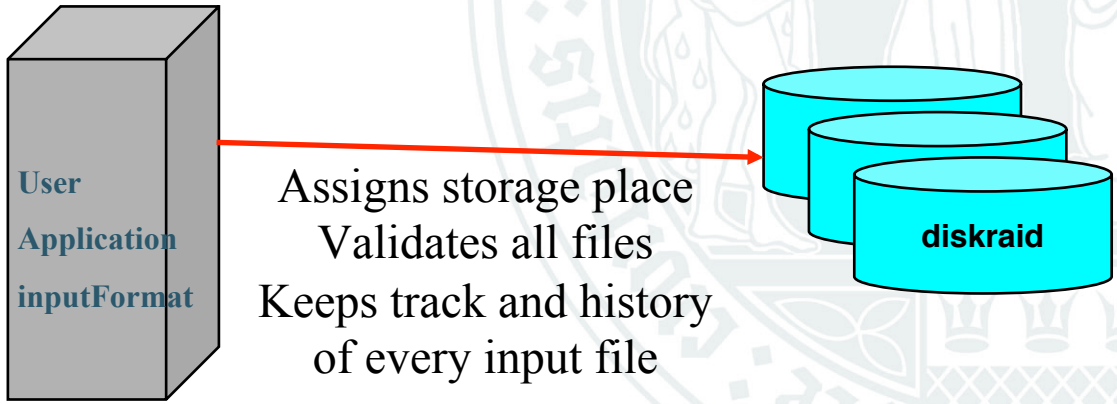
Data flow


Client
Request to store media file, (filename+path, user, topnode, process-xml)
Mediaserver returns a pointer to the mediafile hub
Media-server
Based on user, format, topnode, Media-server validates, stores file, and parses the process-xml to create requested new versions
diskraid

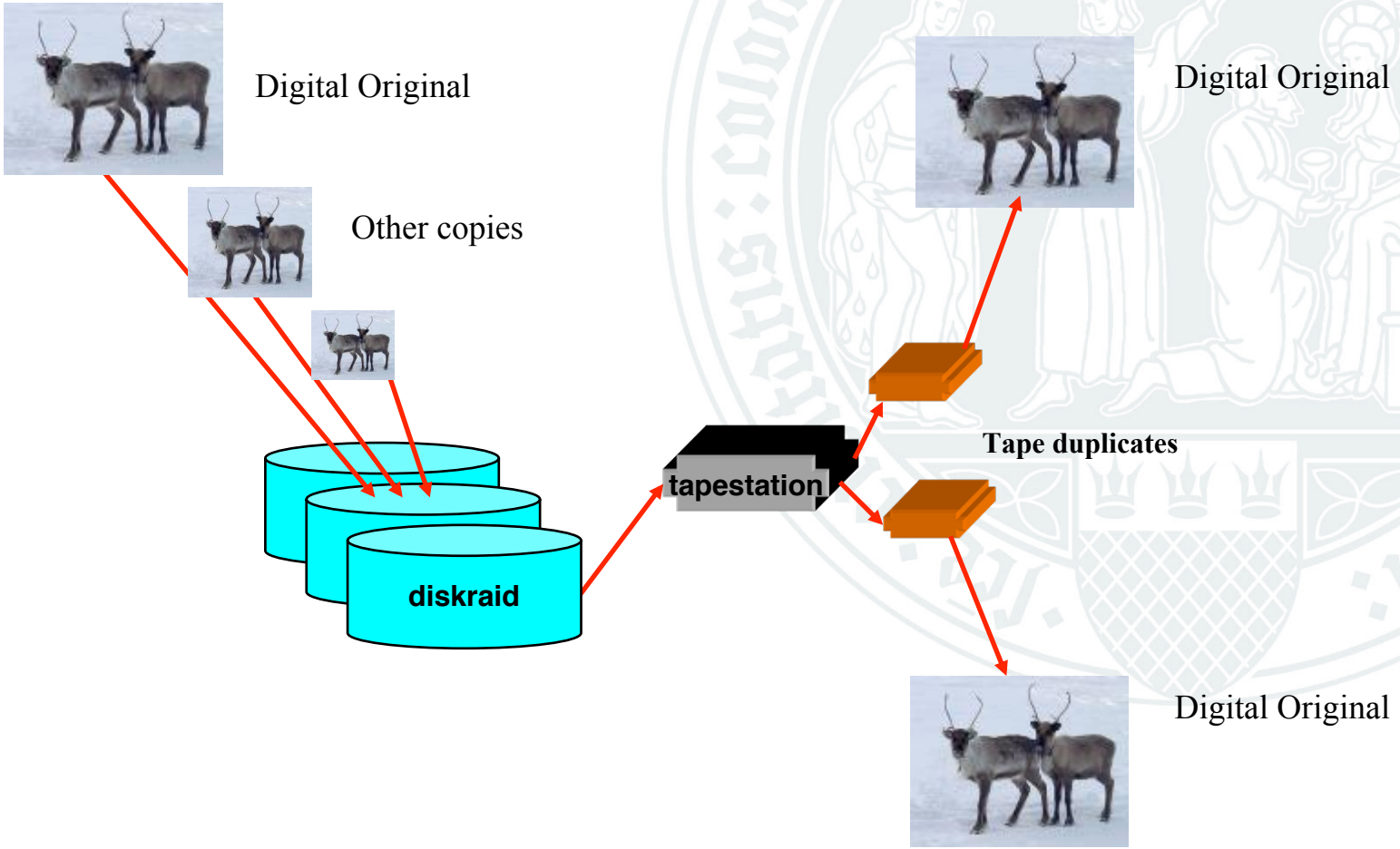
User applications

- A GUI user application is a frontend for:
  - cataloguing pictures (metadata)
  - importing pictures
  - changes and updates
- A command line application is a frontend for:
  - running import scripts
  - file list as parameter
  - meant for expert users
  - meant for large volumes
  - metadata as XML files
  - can link to pre-existing metadata
- Always connected to one discipline schema

Storage keeper



Long term preservation



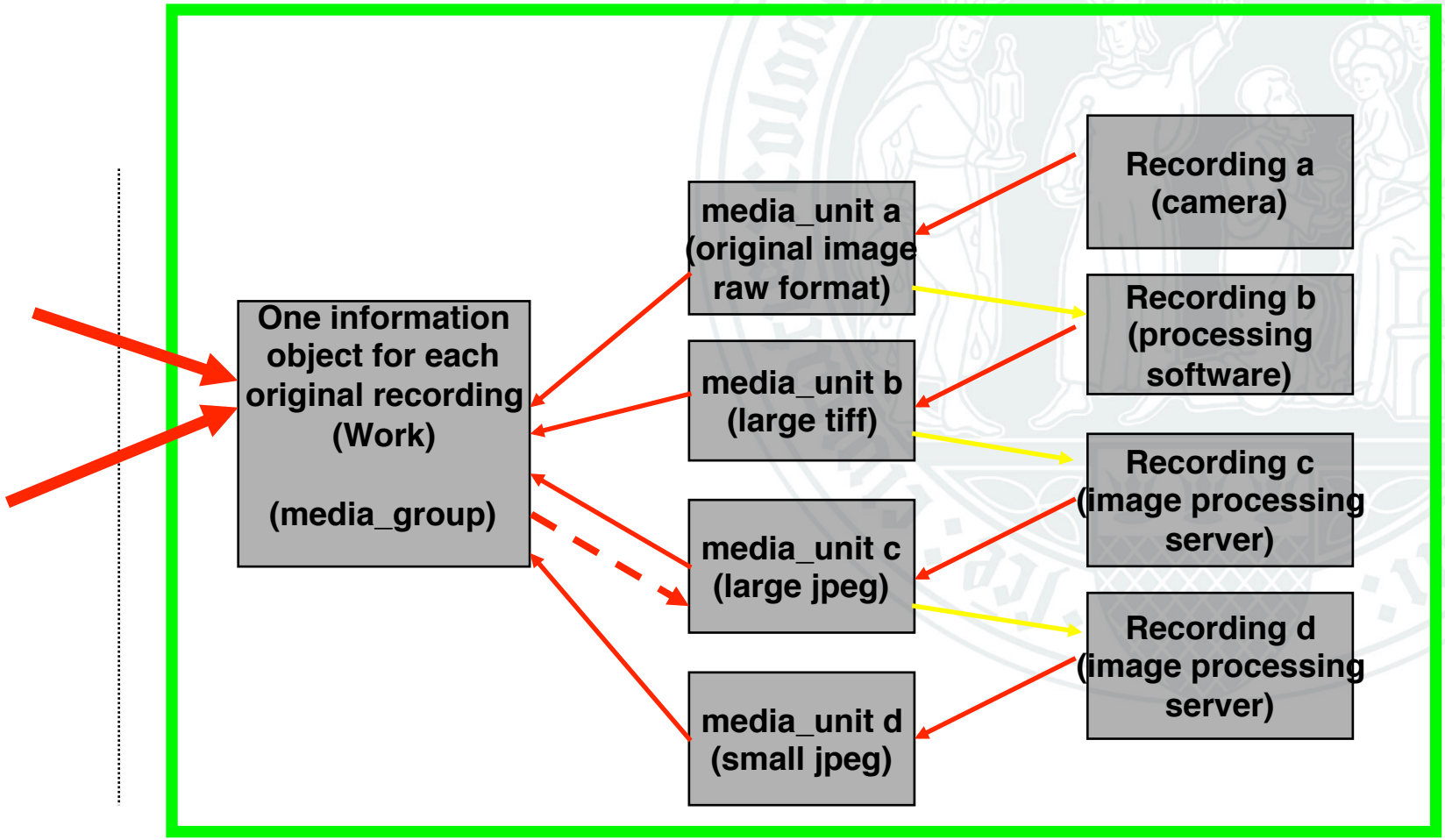
Database

Discipline schema 1
• image_card – media_group_id – concent_metadata • *subject specific tables*

Common schema
• media_group – id – process_xml – tech_metadata • media_unit – tech_metadata – default {0, 1} • schema_setup – process_spec – delete_limit • process – process_spec – status {0, 1, 2, 9}

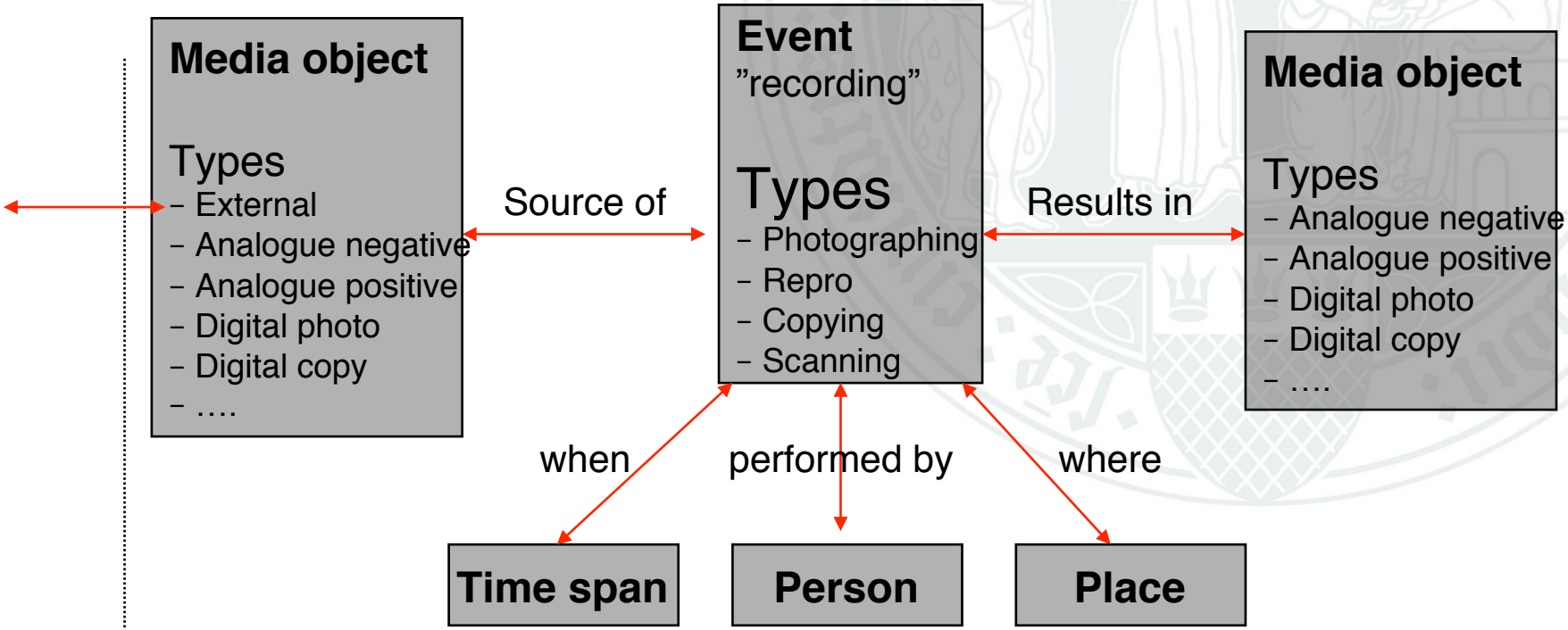
Discipline schema 2
• image_card – media_group_id – concent_metadata • *subject specific tables*

Example work flow (digital image)



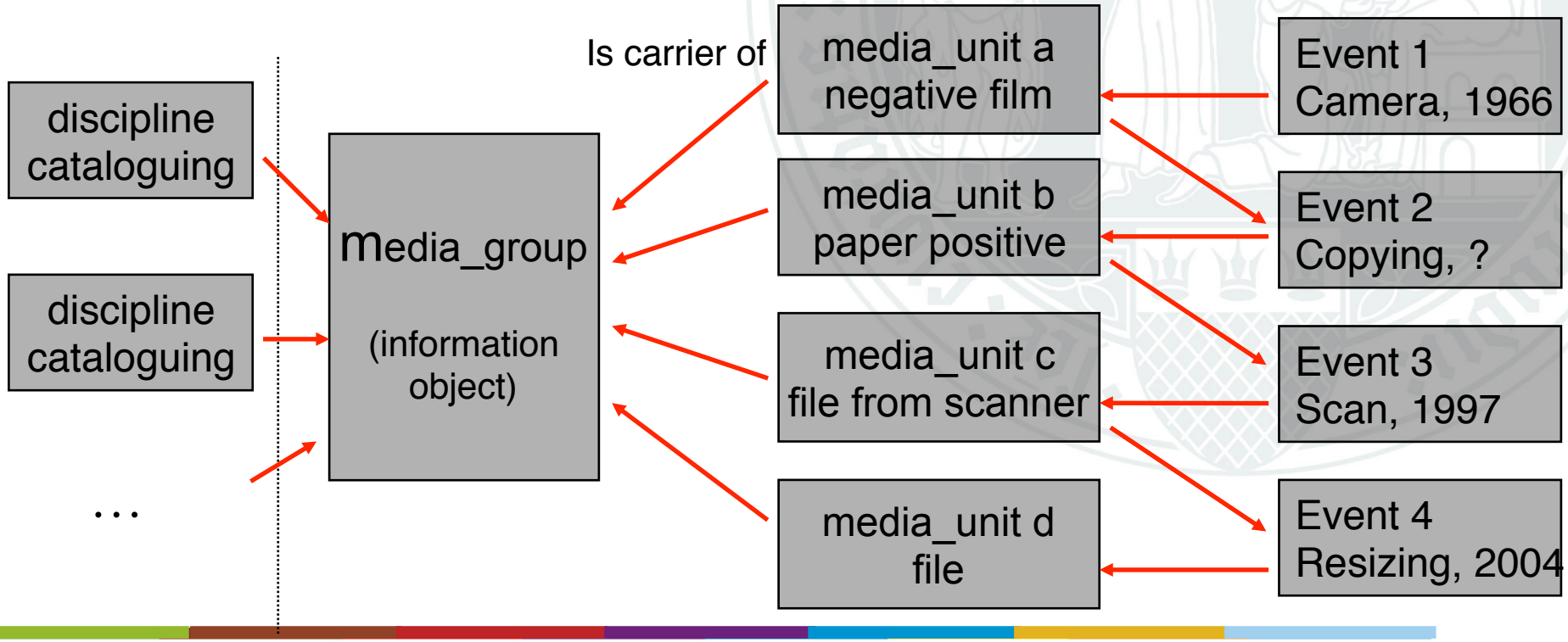
Event centric data model

source → recording → result



Data model example: digitised image

Separation between “information object” and “information carriers”



File processing

- Parse processing XML
- Set up production line
  - any conversion path with possible conversions can be added
- Matrix of in and out formats and default scripts
  - can be overridden.
- Scripts run in background
  - queue handling
  - load balancing

Monitoring

- Zombie jobs
- Server load
- Memory consumption
  - leakage
  - fine on a PC, but with uptime > 100 days…
- Database
  - error messages
  - instability
  - abnormal behaviour
- Must be monitored by humans (using tools)
  - email messages with control data

Extending metadata

- Some metadata must be kept
  - e.g., process records
  - format details
- Some metadata can be changed
  - classification
  - motive description
- But keep old versions
  - institution history
  - legal liability
    - historical institutional/governmental racism
    - land rights