Iterative Analysis of Pages in Document Collections for Efficient User Interaction

Joseph Chazalon, Bertrand Coüasnon, Aurélie Lemaitre

Rennes, Brittany, France

www.irisa.fr/imadoc

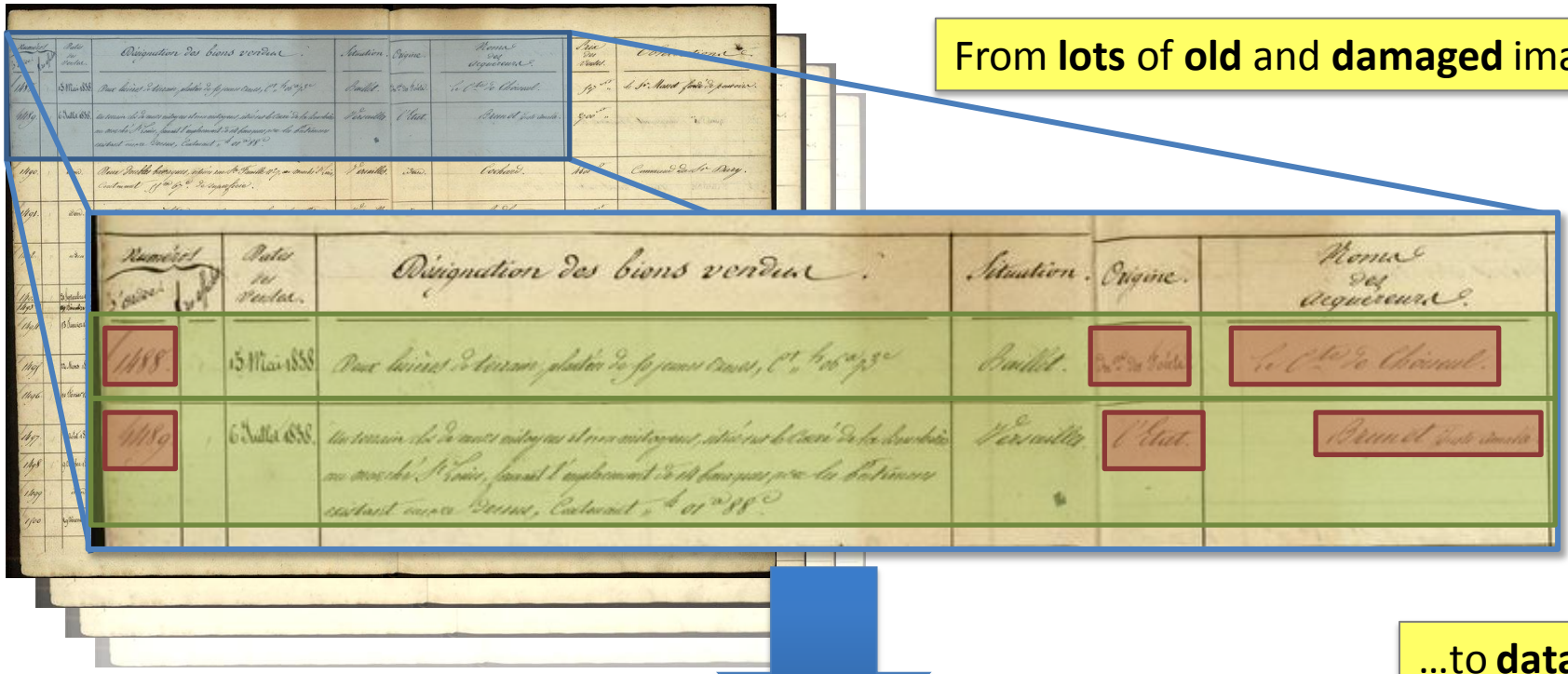
Project funding

Employers

Research unit

Regional Research & Education Network

We extract, recognize and index contents.

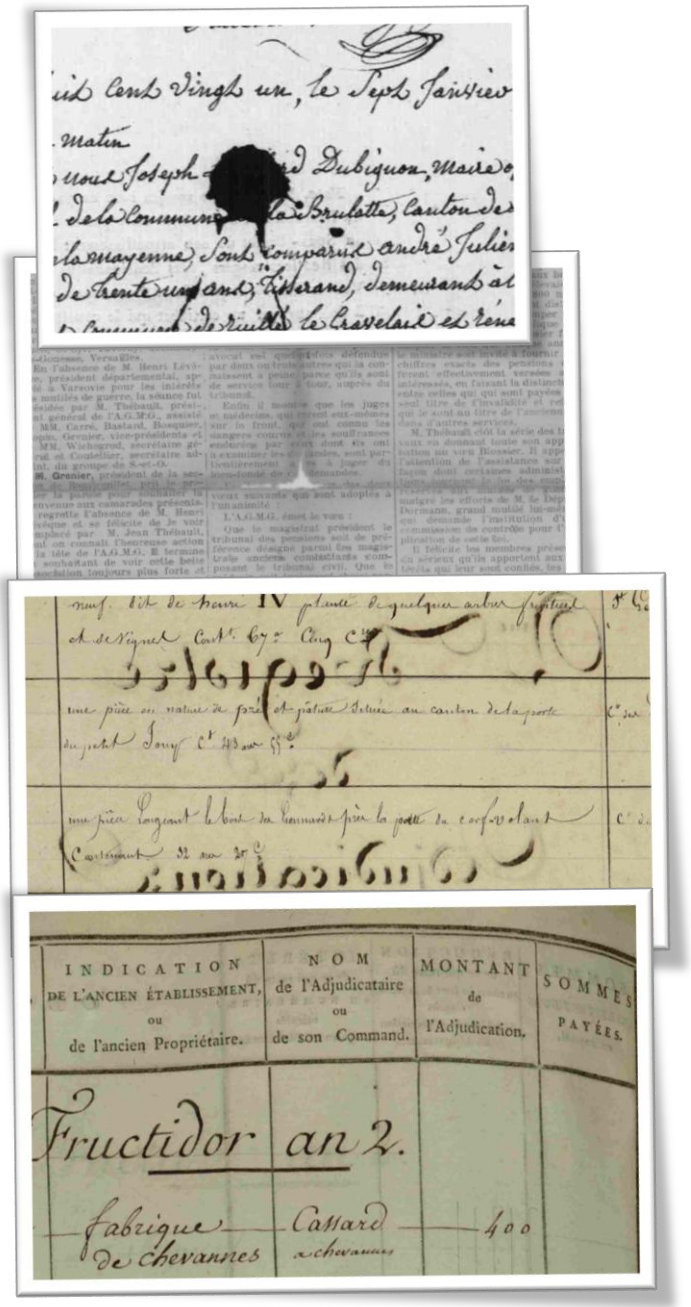


From **lots** of **old** and **damaged** images…

…to **database**.

image_file	sale_number	prev_owner	new_owner
00071.jpg	1488	???	Cne Choiseul
00071.jpg	1489	Etat	Brunet

1/ Document alterations



Composition

Storage
Transformation
Edition

Digitization

During document lifetime, alterations (+/-/~) of

- Structure
- Contents

Document = noisy channel

Perturbation of

- Analysis
- Recognition

even with well-defined document models

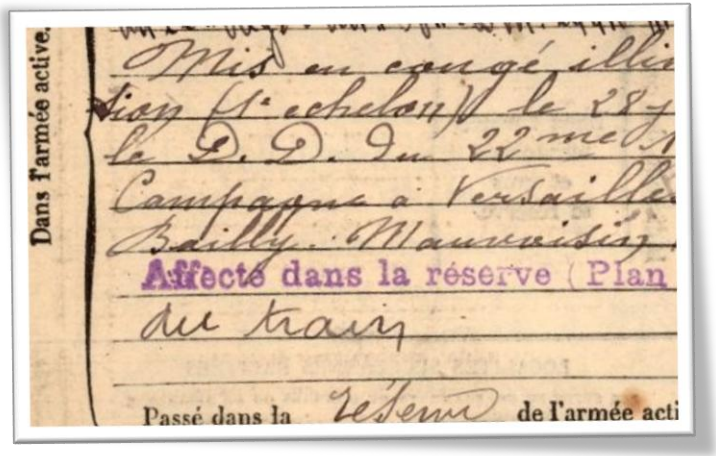
2/ Document variability

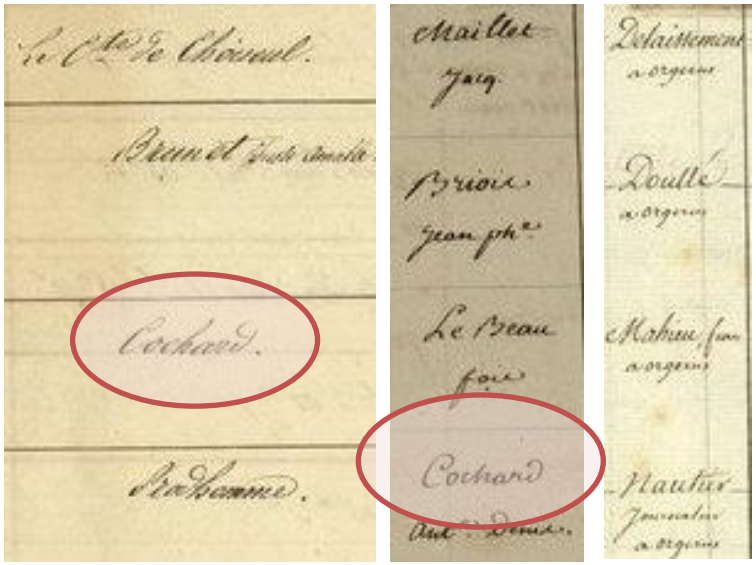
Unexpected things will happen

- Important amount of pages
- Writing style can change
- Content can be mixed
- Structure can change
- Open vocabulary

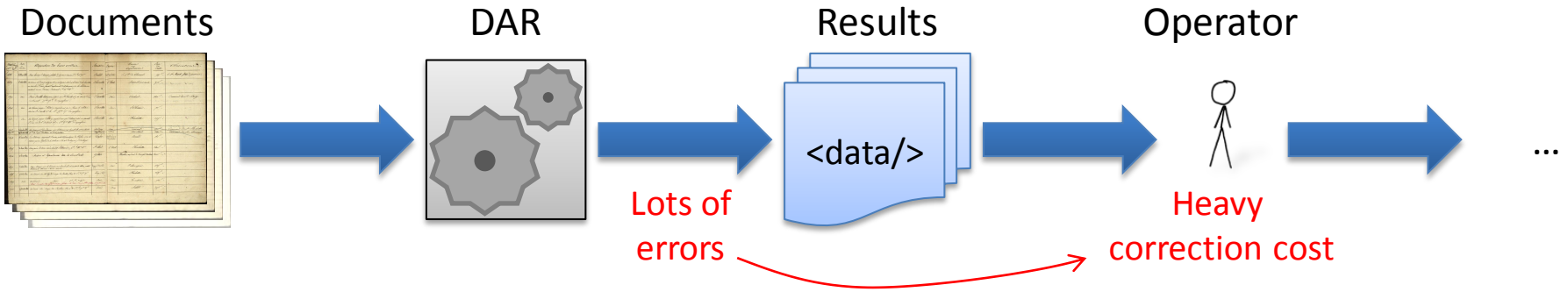
Document models are not perfect

- Lack of ground truth
- Many special cases





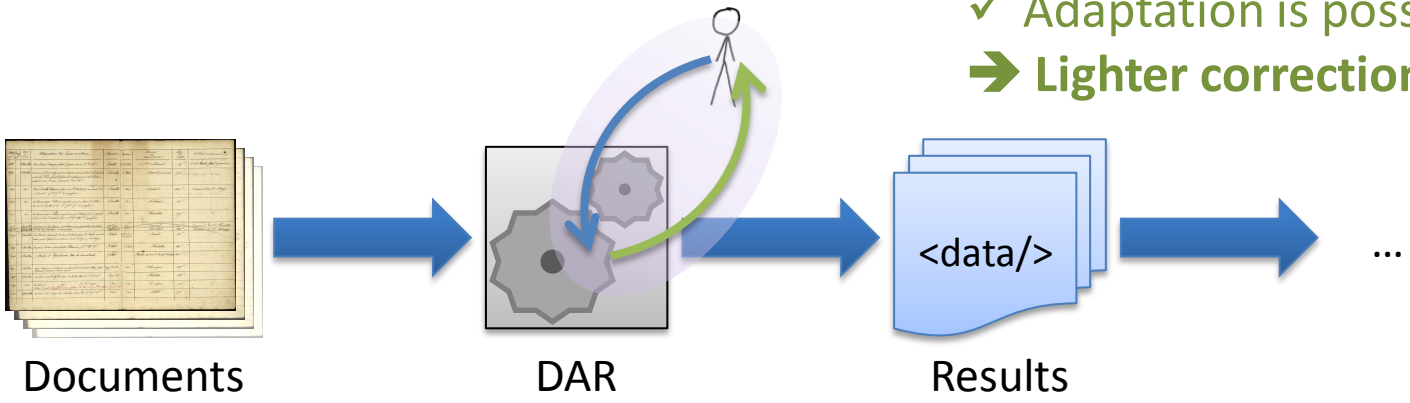
We need human interaction



A **linear analysis workflow** leads to an **inefficient interaction** (in our case)

We need to **correct, guide, enrich** the system **during** the analysis stage

- ✓ Early error correction
- ✓ Limited error propagation
- ✓ Adaptation is possible
- ➔ **Lighter correction cost?**



Efficient interaction during analysis stage

Goals

1/ Durable influence of information provided

- Correct previous results
- Improving later processing

2/ Asynchronous interaction model

- Human operator & DAR system must not wait for each other

3/ Keep document models simple

- No time-related considerations
- No information flow description

Outline

A/ Architecture

➔ Iterative analysis of pages

B/ Implementation of a page analyzer

➔ Extension of document model language

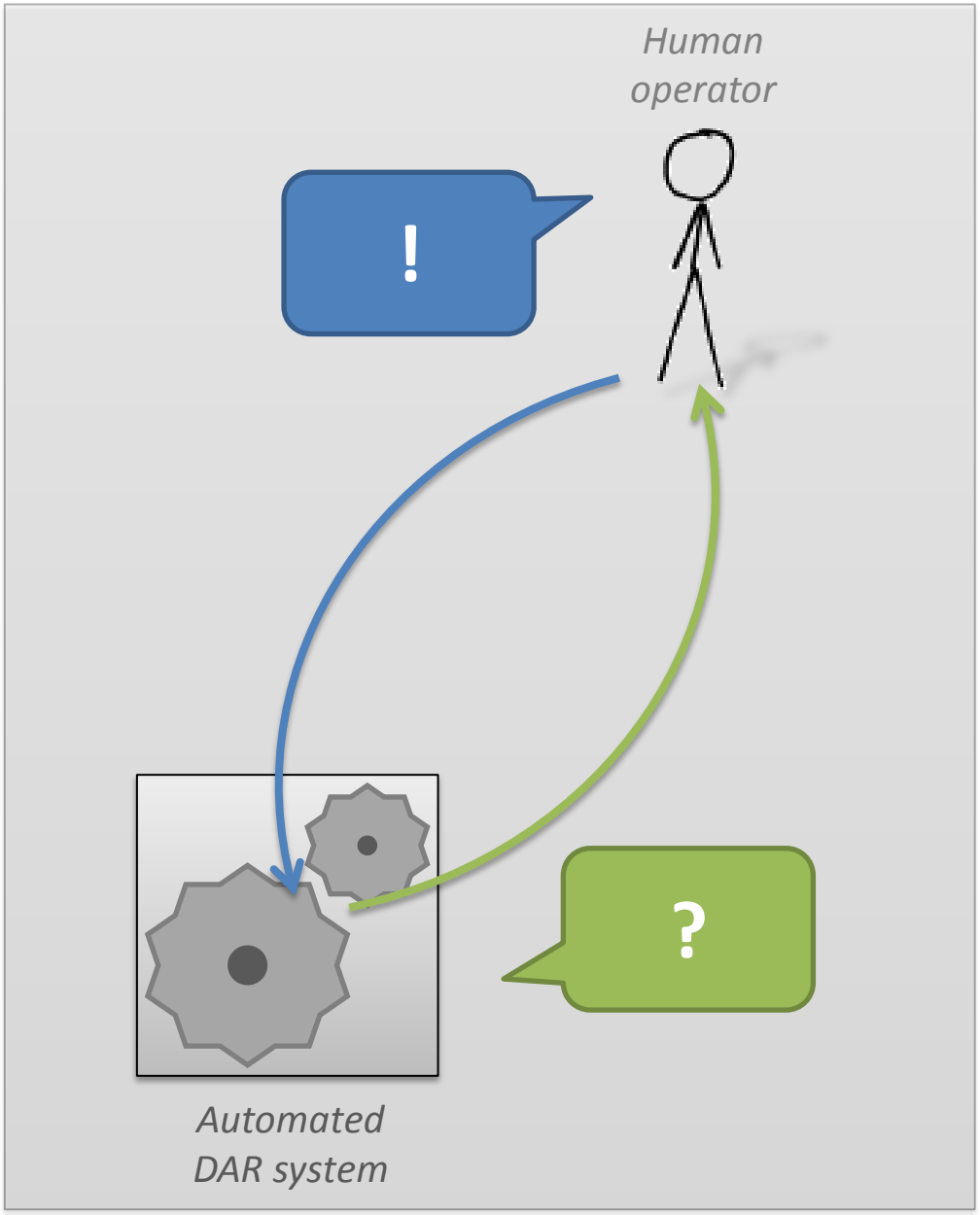
C/ Validation

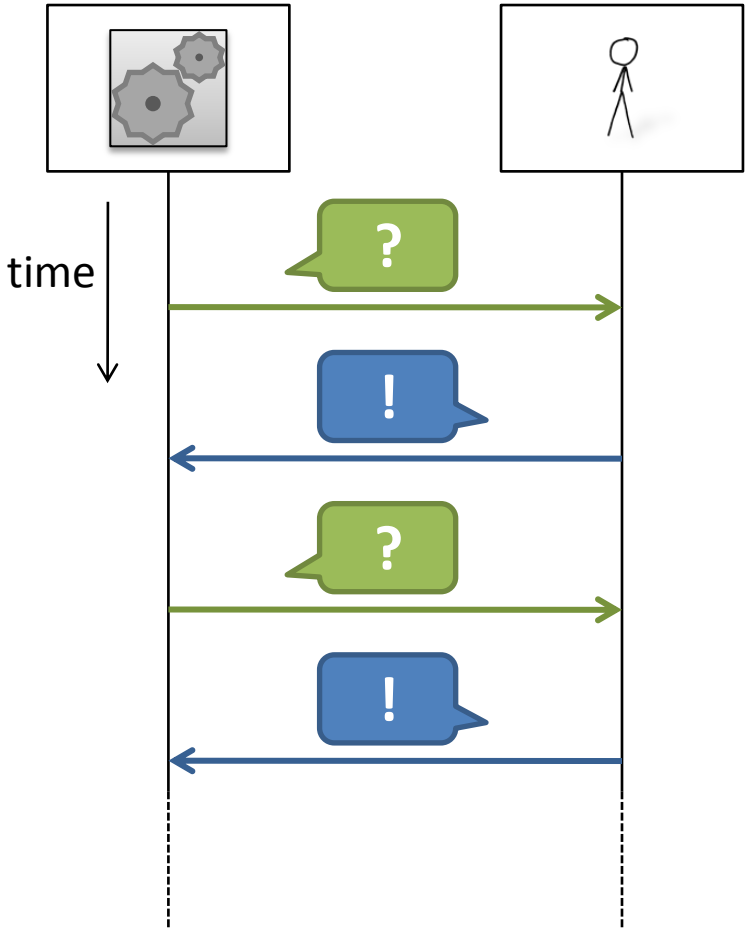
➔ Application on real cases

How can we **enable an efficient interaction** during the analysis stage?

ARCHITECTURE FOR AN ITERATIVE ANALYSIS

Directed interaction model

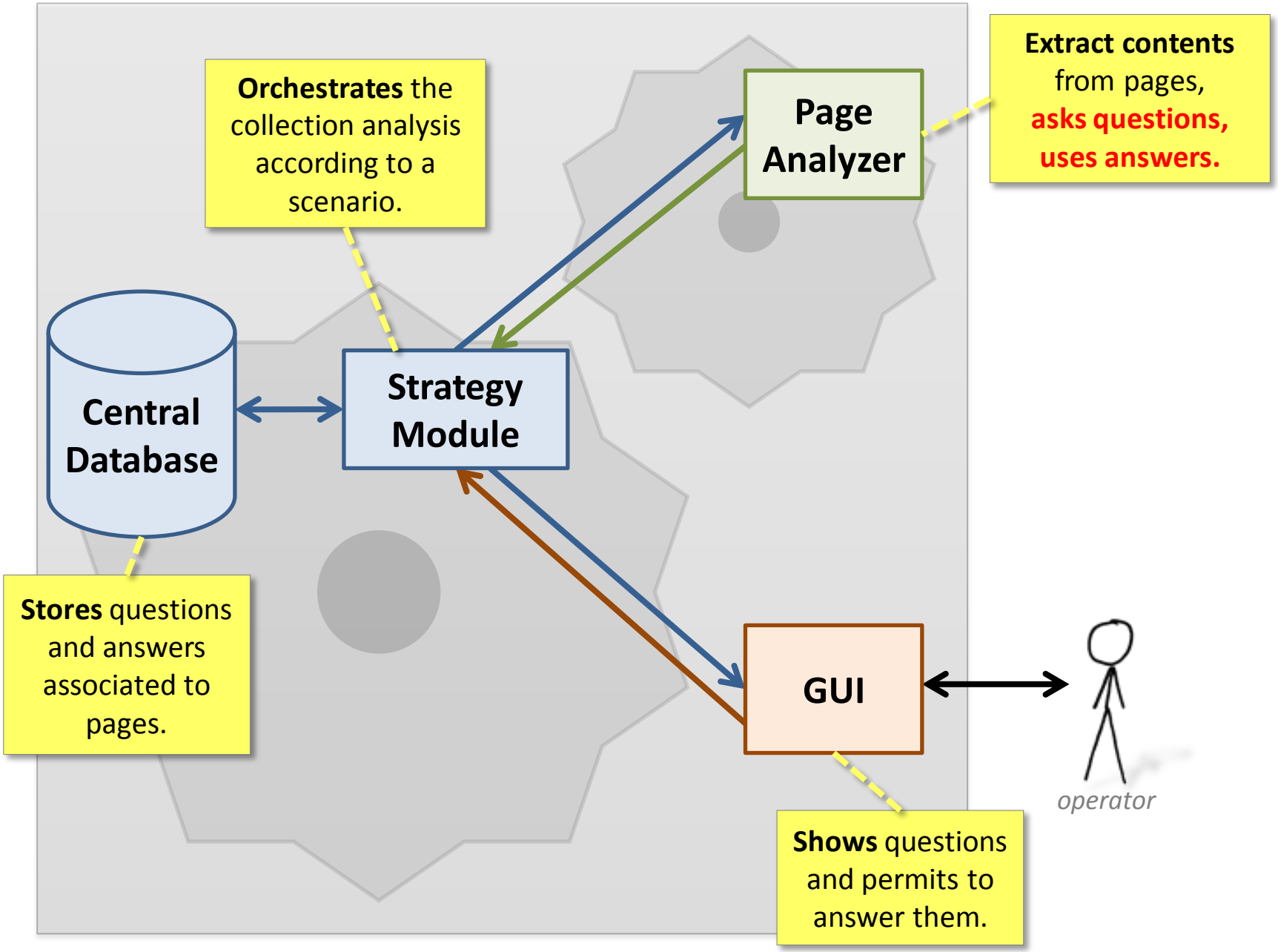




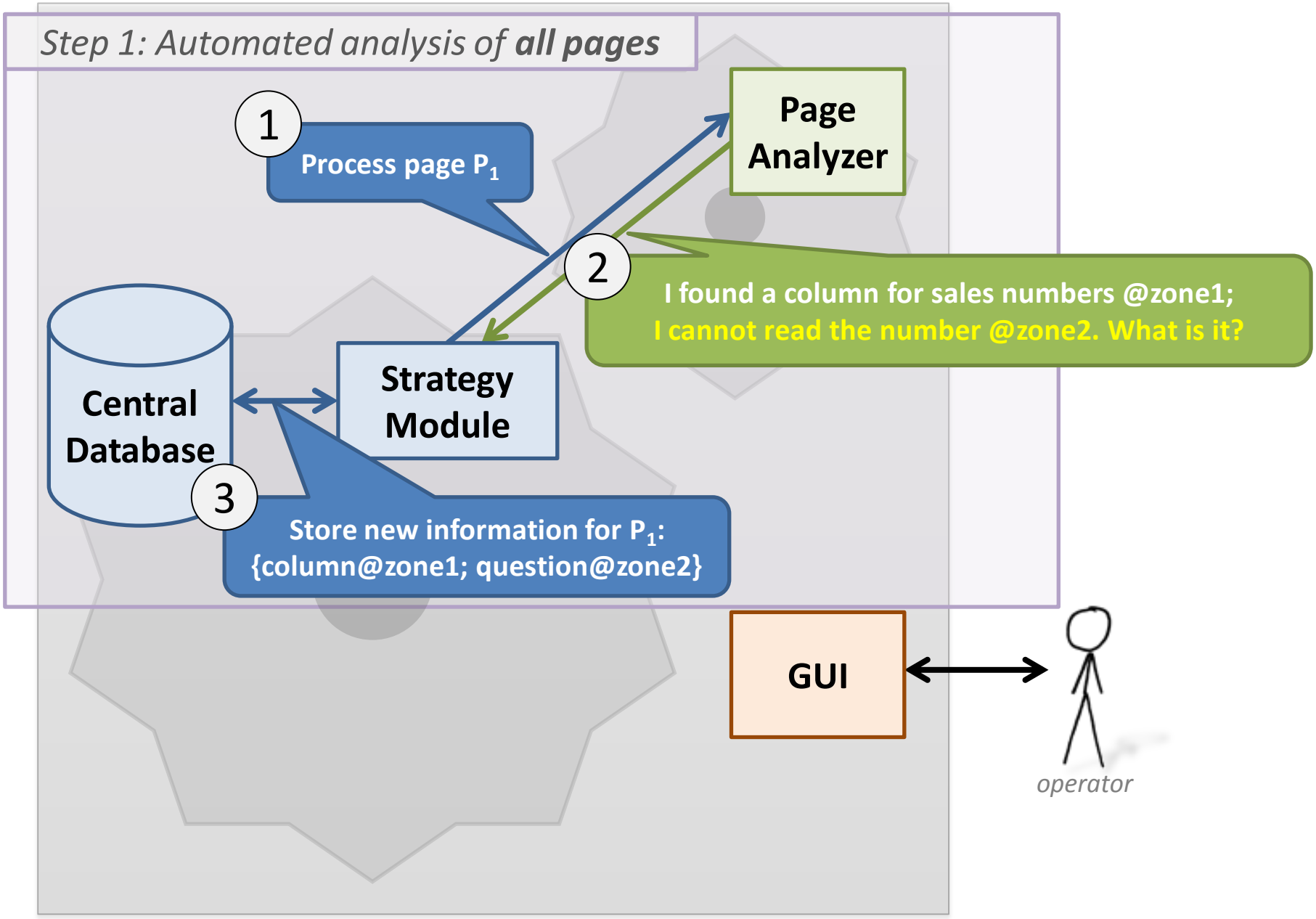
The system

- ***asks questions*** *to a human operator*
- ***uses the answers*** *to progress*

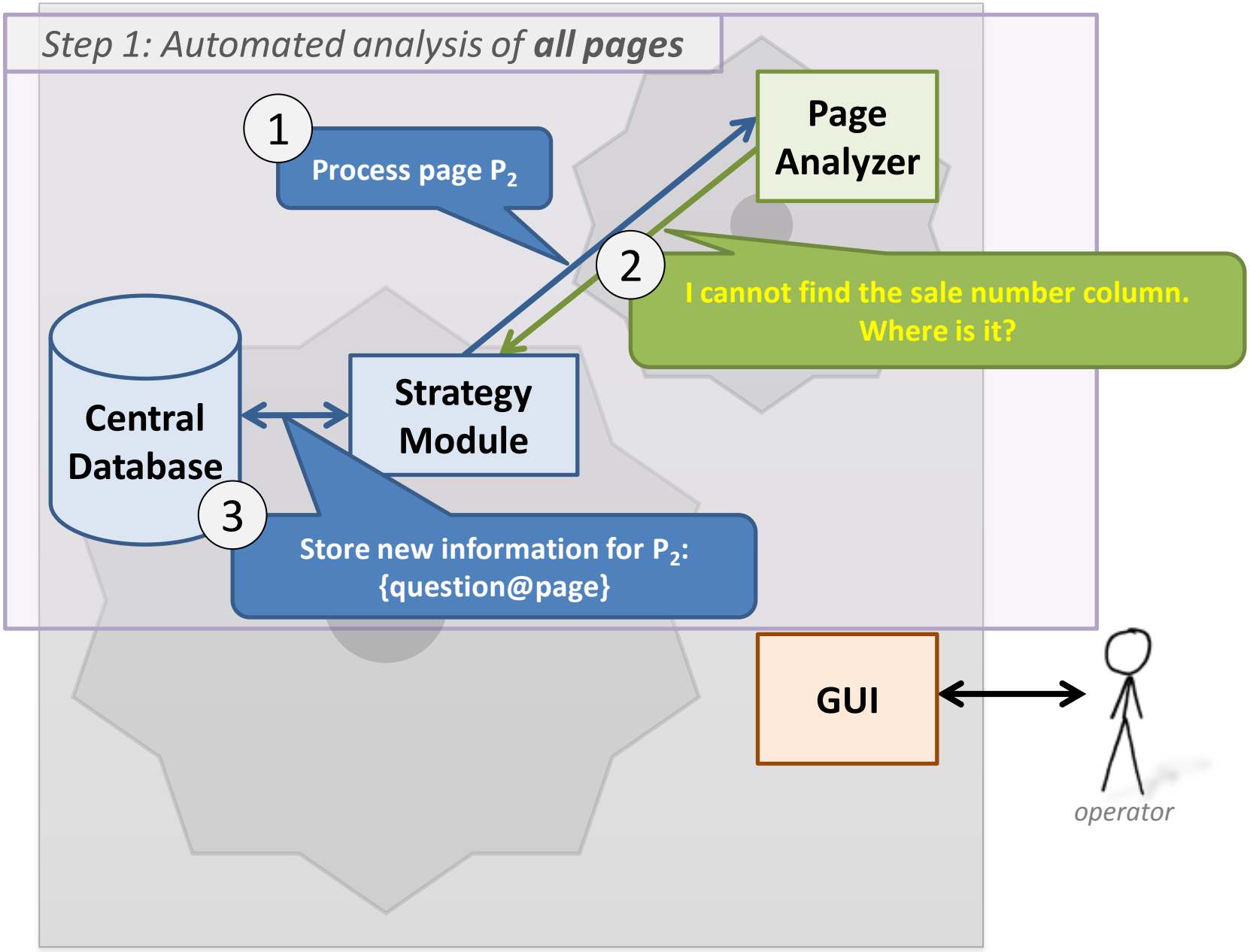
Required components



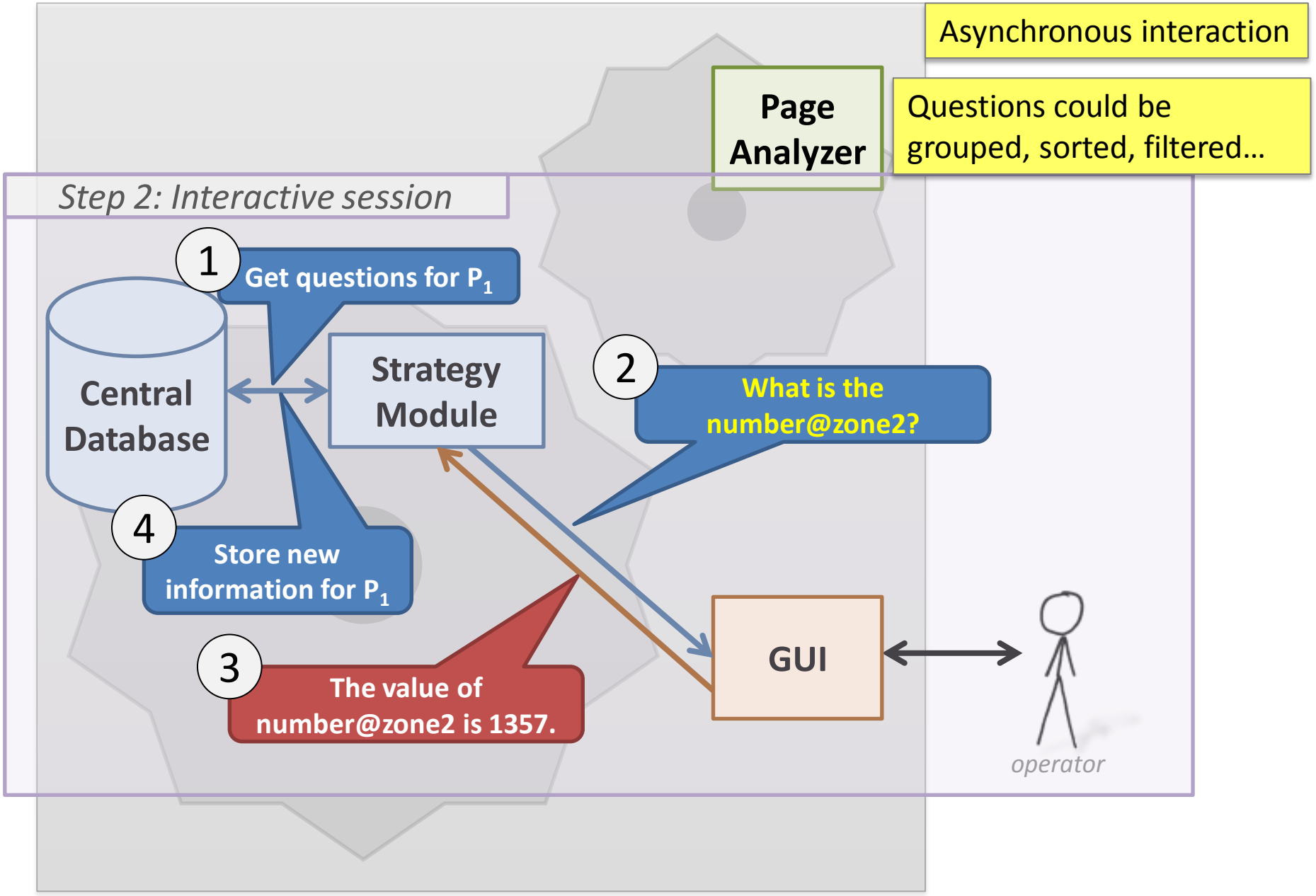
Iterative analysis behavior: example



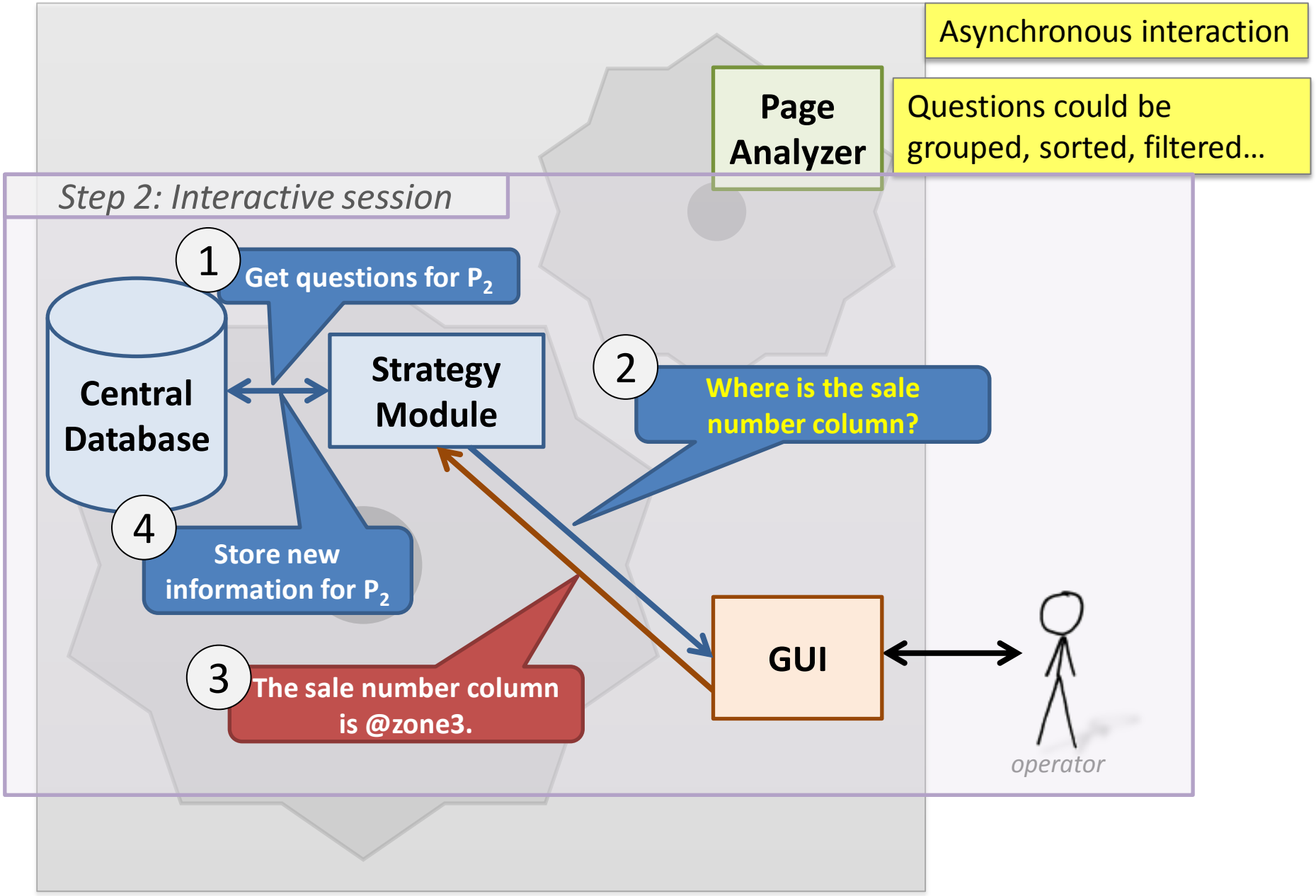
Iterative analysis behavior: example



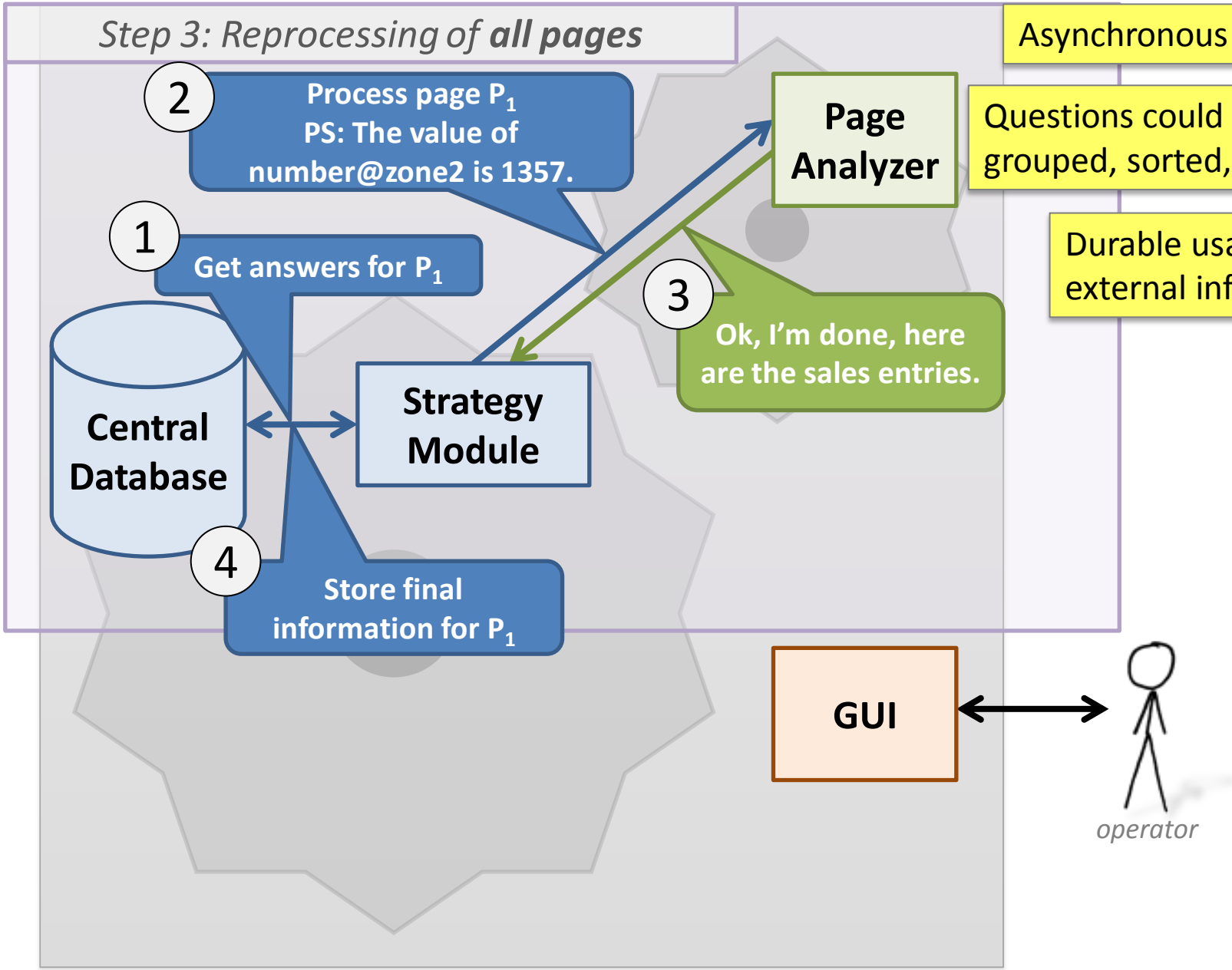
Iterative analysis behavior: example



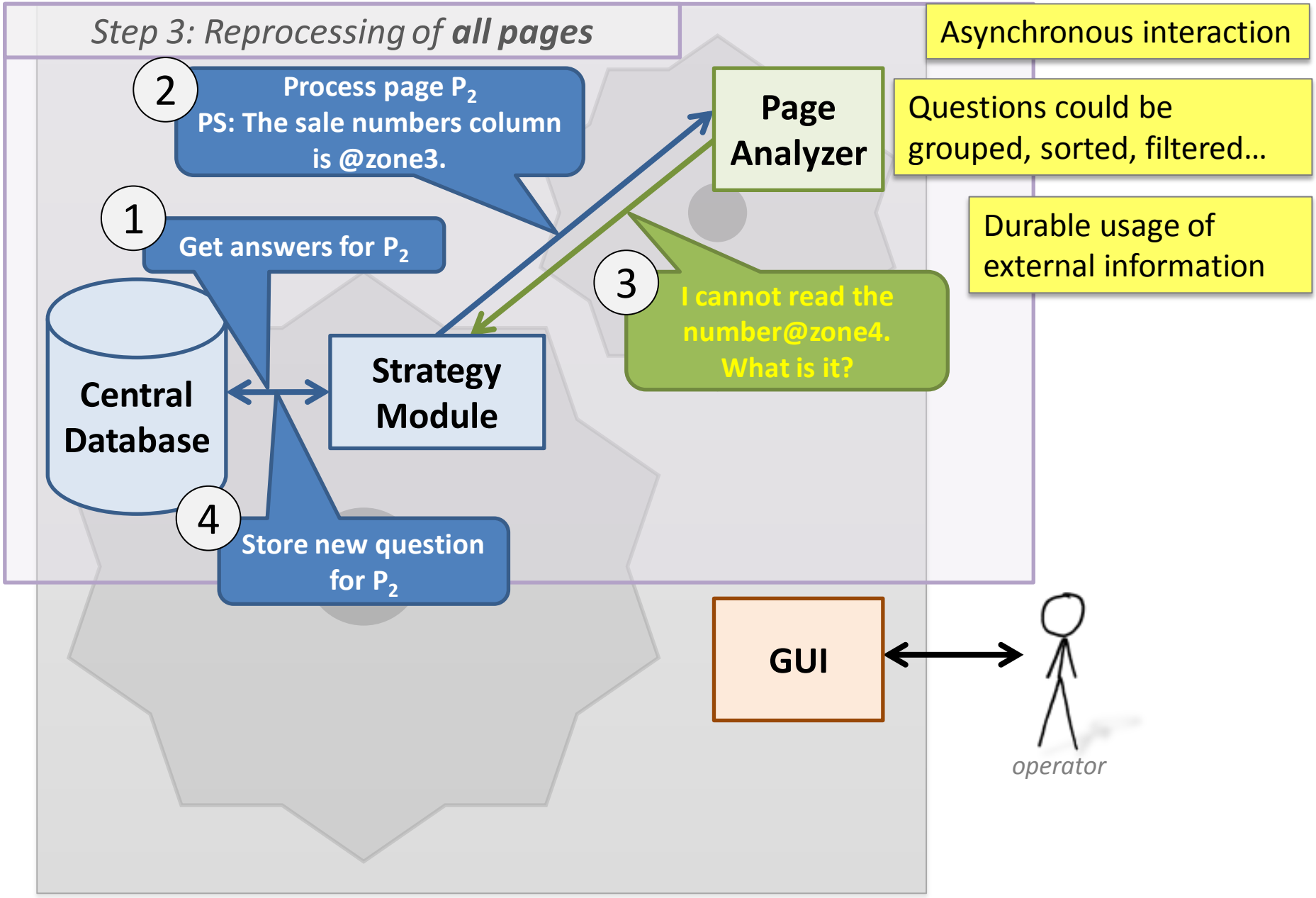
Iterative analysis behavior: example



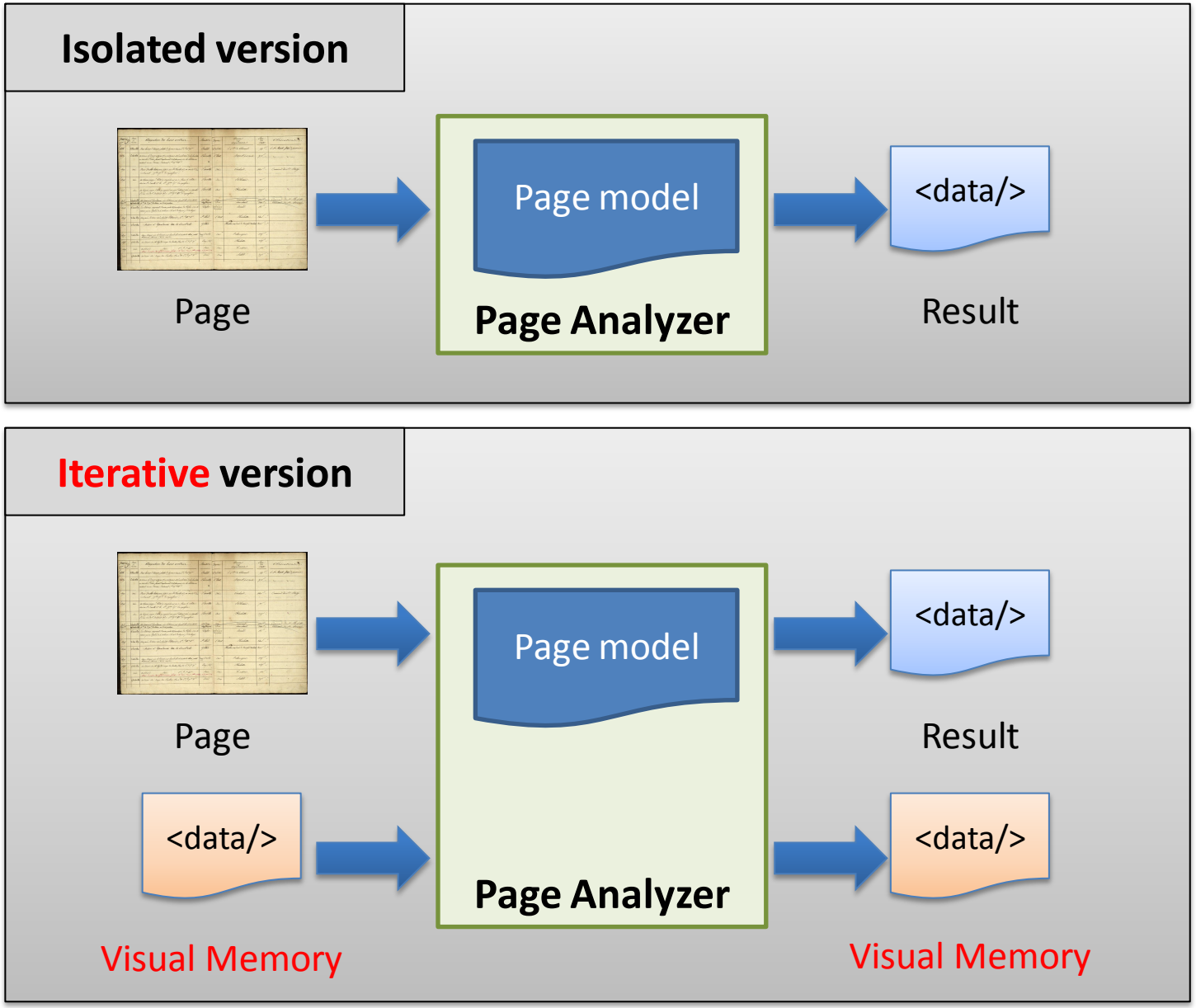
Iterative analysis behavior: example



Iterative analysis behavior: example



Focusing on the Page Analyzer



How can an **isolated page analyzer** be turned into an **iterative one** and enable **interaction**?

IMPLEMENTATION OF AN ITERATIVE PAGE ANALYZER

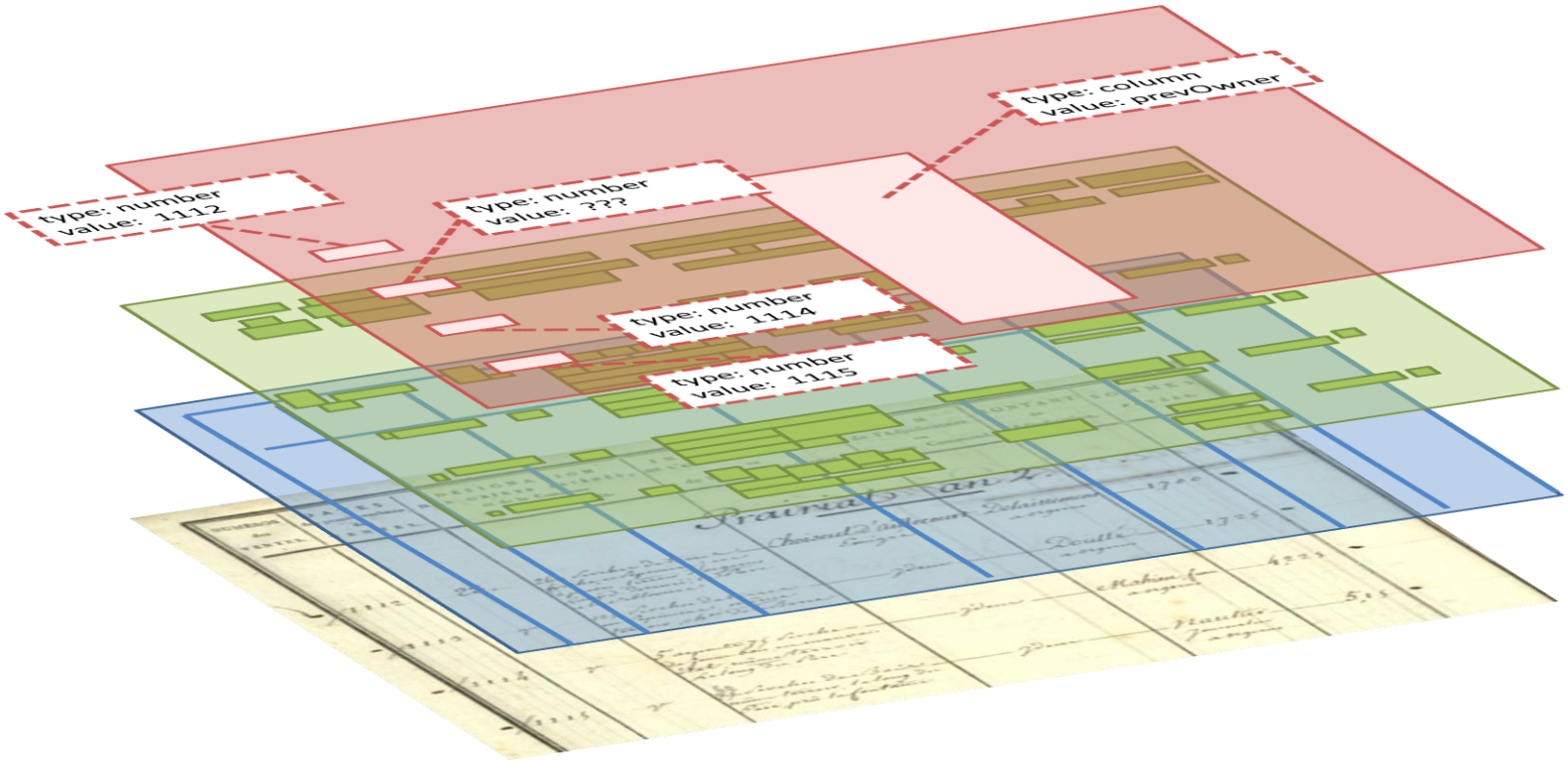
Visual Memory & Page Analyzer

We use a layered structure for page analysis

External information uses one more layer

This *Visual Memory* has 3 properties:

1. Same referential as the image
   each element has a shape + a position
2. Information is available at any moment
3. Same access operations as image data
   easy creation, modification, deletion



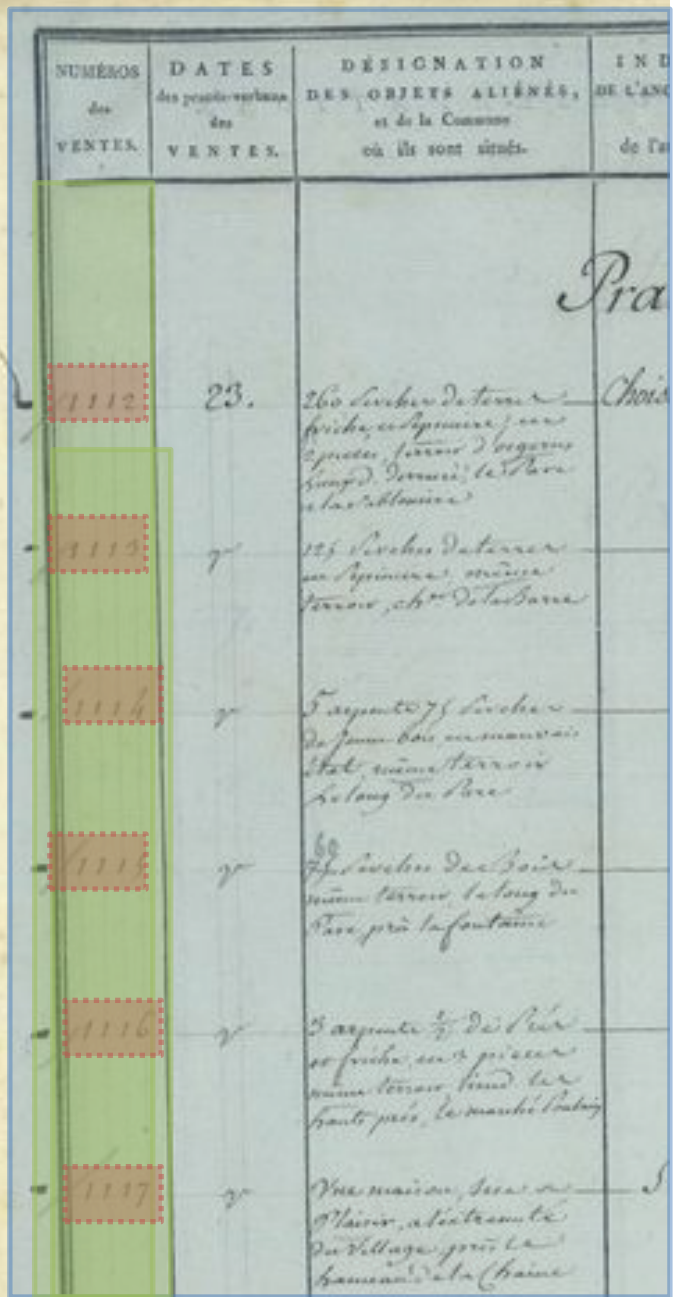
External data =
Visual Memory

Components

Lines

Image

DMOS analysis without interaction




```
start() ::=
    AT(allPage) &&
    locateLeftCol(-ColPos) &&
    AT(+ColPos) &&
    readAllNumbers().

readAllNumbers() ::=
    locateNumber(-NumPos) &&
    recognizeNumber(+NumPos, -Value) &&
    % use Value...
    AT(under -NumPos) &&
    % loop until no more numbers...
```

Enable interaction with iterative analysis

3 Implementation challenges		
Detect problems and **ask questions**	**Use answers** if they exist	**Continue the analysis** in independent parts of the page
Extension of document model language : Asynchronous error		
detection	**correction**	**recovery**
Syntax: 3 new operators for document models ➔ Low impact on document models		
`raiseQuestion(…)`	`getAnswerOrTry(…)`	`catchQuestion(…)`
Implementation: manage execution and information flow automatically ➔ Uses exception-like constructs, easy to adapt to your system		
store question in memory **raise** "interaction exception"	**if** answer exists in memory **then** use this answer **else** try the rule	**continue** the analysis *(catch "interaction exception")*

Without vs. with interaction

Without interaction

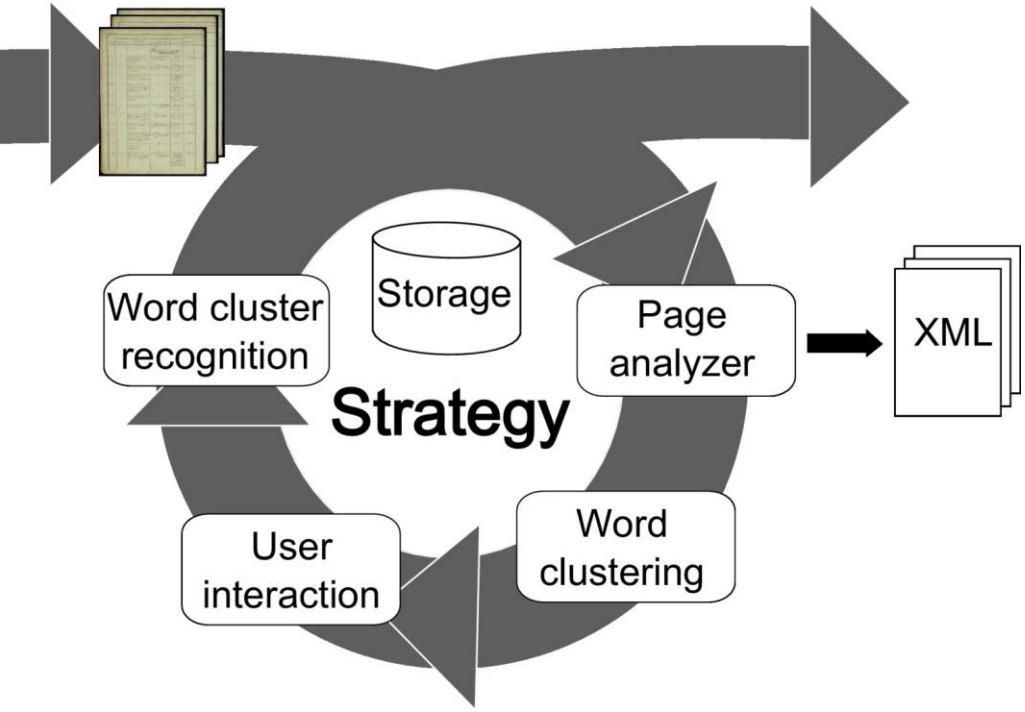
```
start() ::=
   AT(allPage) &&
   locateLeftCol(-ColPos) &&
   AT(+ColPos) &&
   readAllNumbers().

readAllNumbers() ::=
   locateNumber(-NumPos) &&
   recognizeNumber(+NumPos,
                   -Value) &&
   % use Value...
   AT(under -NumPos) &&
   % loop...

locateLeftCol(-ColPos) ::=
   ...

recognizeNumber(+NumPos,
                -Value) ::=
   ...
```

With interaction

```
start() ::=
   AT(allPage) &&
   getAnswerOrTry(colT, -ColPos,
      locateLeftCol2(-ColPos)) &&
   AT(+ColPos) &&
   readAllNumbers().

readAllNumbers() ::=
   locateNumber(-NumPos) &&
   catchQuestion(
      getAnswerOrTry(numT, -Value,
         recognizeNumber2(+NumPos,-Value)
    )) &&
   % use Value (may be uninstantiated)
   AT(under -NumPos) &&
   % we still can read other numbers...

locateLeftCol2(-ColPos) ::=
   locateLeftCol(-ColPos) ##
   raiseQuestion("Where is the column?",
                 allPage, colT).

recognizeNumber2(+NumPos,-Value) ::=
   recognizeNumber(+NumPos,-Value) ##
   raiseQuestion("What is this number?",
                 +NumPos, numT).
```

VALIDATION

Iterative analysis of pages in a collection



Exploiting Collection Level for Improving Assisted Handwritten Word Transcription of Historical Documents

Poster #48, session 2
Today @ 13:40-15:20

We used

- Iterative page analysis
- Handwritten word **clustering**
- Handwritten word **recognition**
- Human interaction

And tested several scenarios

➔ **28% reduction** of human workload for "indexation" scenario

We were able to

- **Reduces recognition error rate**
- **Reduce human workload**

CONCLUSION

Conclusion

Goal: Efficient interaction during the analysis stage

- Durable effects
- Asynchronous
- Low impact on document model

Efficient interaction requires an architecture with

- GUI
- Strategy module
- Central database
- Iterative page analyzer

Easy adaptation of your system to an iterative analysis

- Visual memory structure
- 3 new operators
  - Exception handling provides a simple way to implement them

Approach validated on several tasks

More details at poster panel #48 after lunch