# Orde! with Rexx

Michiel H. van Hoorn

**A concept and tooling for structurering of information and for knowledge management**

# Agenda

- Introduction
- The Orde! approach
- The role of Rexx
- Demonstration

# Introduction

- Mathematician,
  . PhD (1983), thesis on queueing models
- Got lost in IT since 1984
  . Cap Gemini, IBM, Dreamware BV

- Many IT architecture projects
  . management of PC networks in large organisations
- Each project more or less the same
  . working procedures
  . struggling to write consistent documents in time

- and in later project phases
  . bad reuse of work already done
    . a lot of cut and paste
    . loss of consistency

# Idea of Orde!

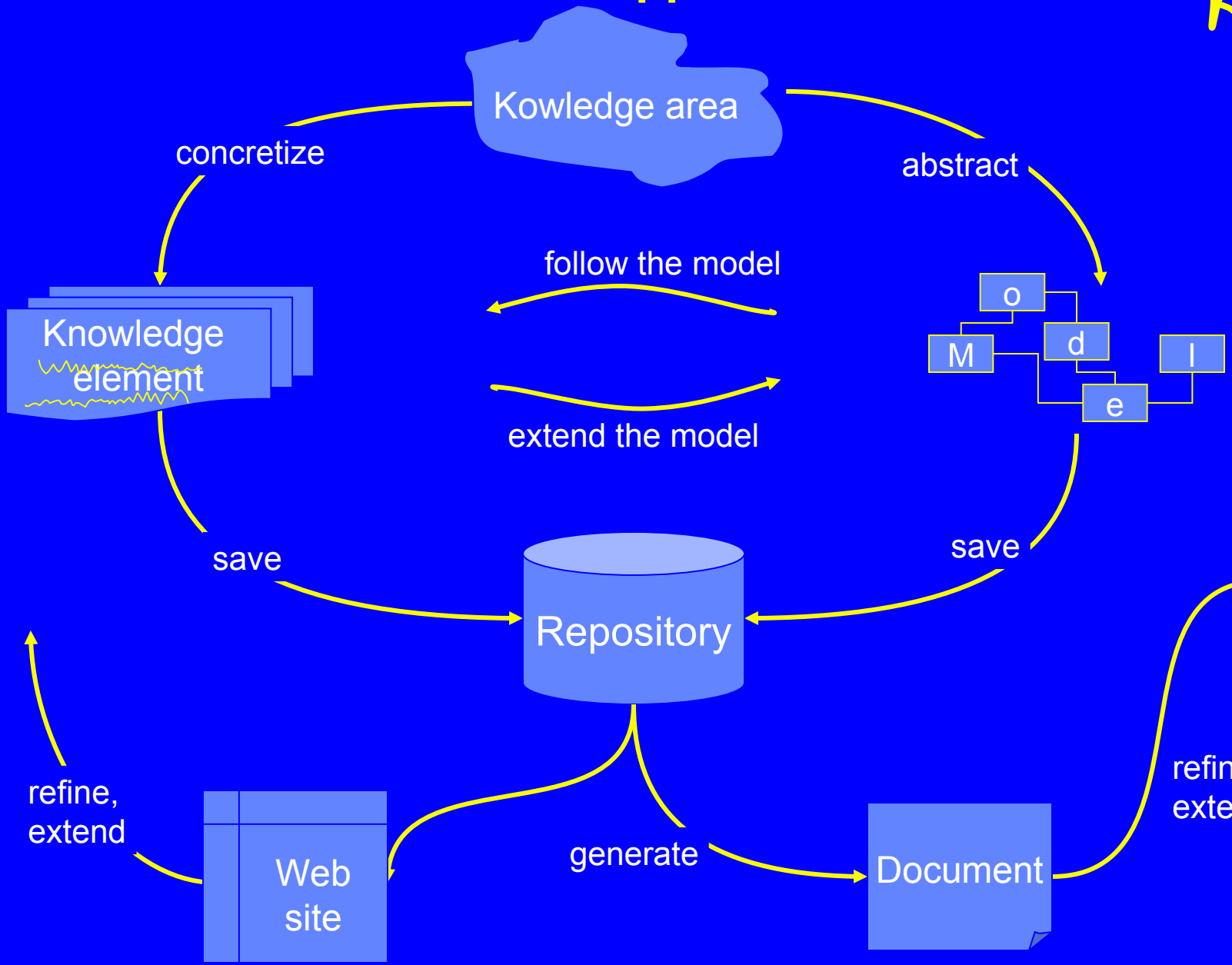
## In a project

- Collect information
- Bring structure in information
- Store in a database
- Generate baseline documents

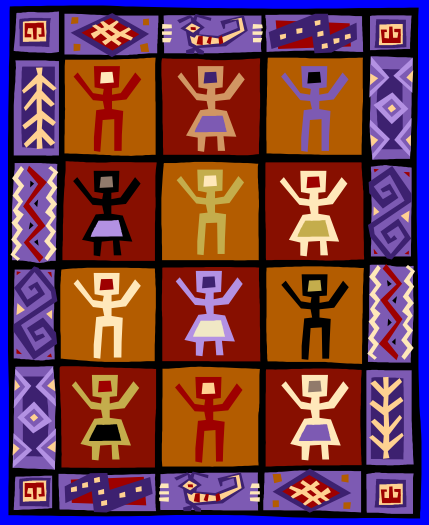
## In next phase of project

- Build on work already done
- Reuse and go in more detail

# Orde! approach



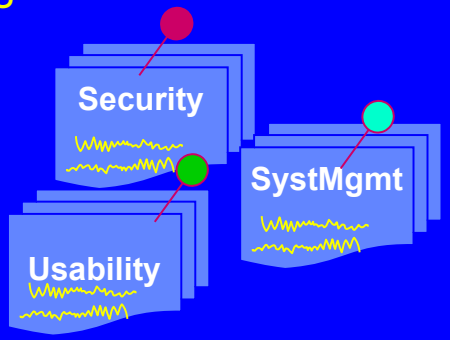
# Knowledge Area



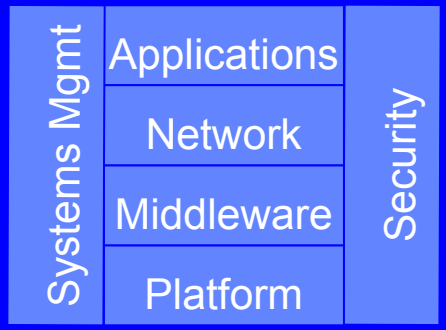
Stakeholders

A knowledge area is field of interest for some reason.

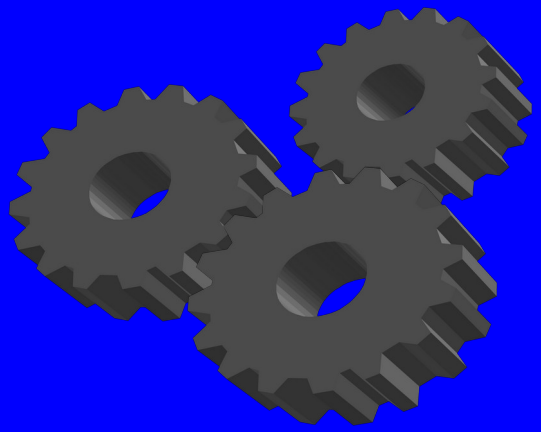
For example: All information gathered during an IT Infrastructure Design project to develop a PC infrastructure



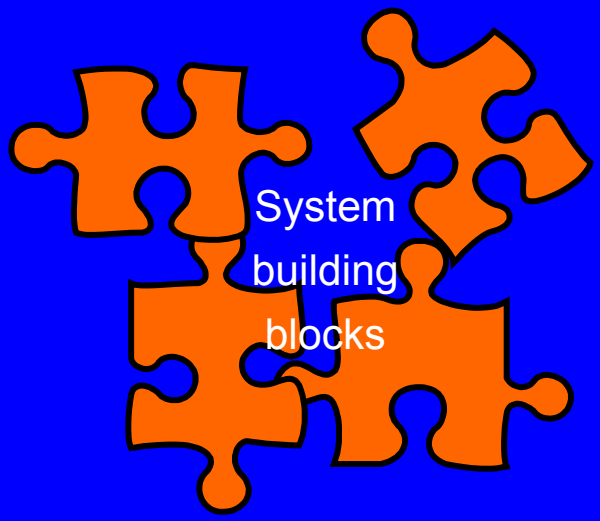

Requirements




Global system solution



Components

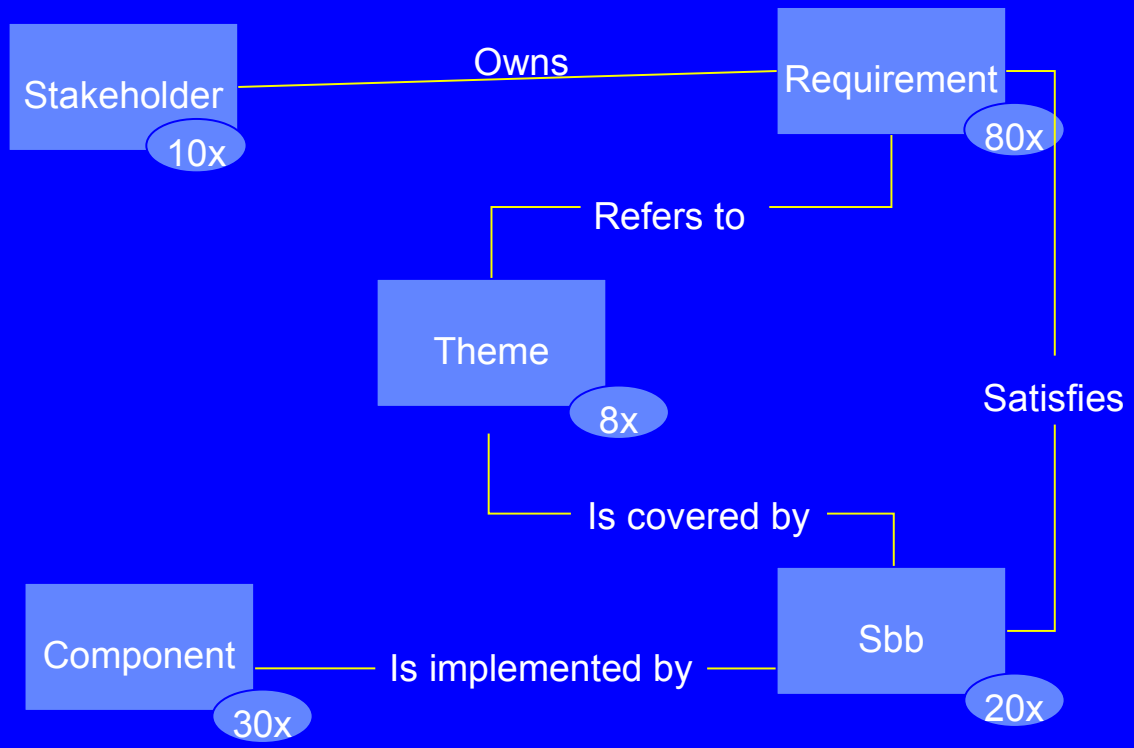


# Model

"The model is a classification of the information gathered."

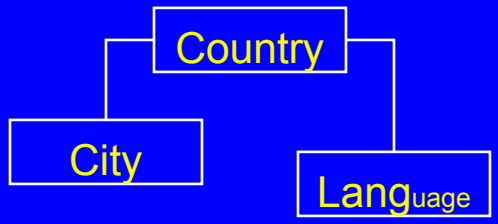
Often it is quite obvious which classes you should distinguish.

The terminlogy used by Orde!
is that of entitity relationship
modeling
- entities
- attributes
- relations



# Knowledge element (1/2)

A knowledge element is a unit of information
- belongs to a certain entity
- has an unique Id
- and also a Name
- has scalar attributes (type: text)
- has relational attributes (type: list of Id's)




[Country = NL]
Name = The Netherlands
Population = 16.000.000
Description =
<Name> is a West-European country on the North Sea. It's capital is [City.Amsterdam] and the government is seated in [City.TheHague].
<Name> has <Population> inhabitants; <City.Amsterdam;Population> of them live in the capital.
[Fig.MapNL_s].
Country_Has.Lang = Dutch Frysk

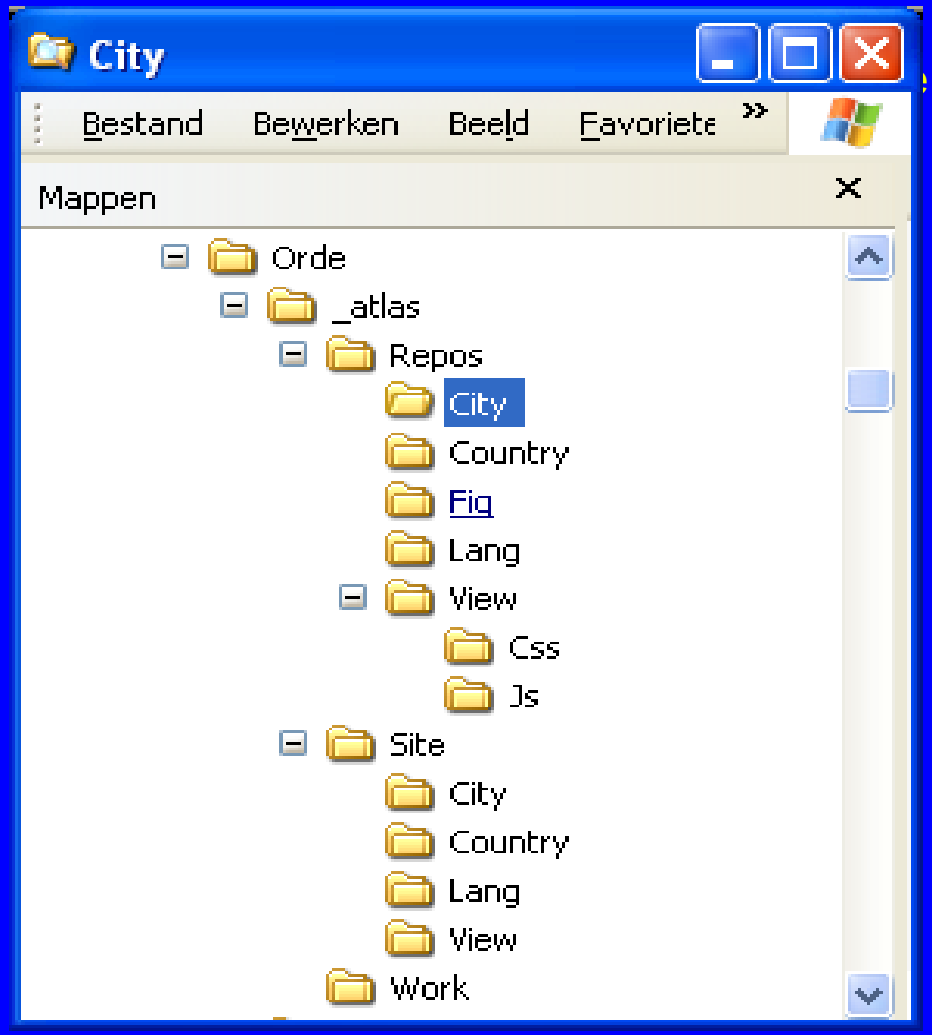
# Knowledge element (2/2)

A knowledge element can be stored in
- a .sim file as text
- a .csv file in tabular form
  (or .xls and translated automatically to .csv)

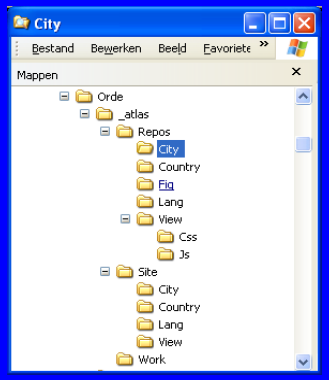
| Ent | City | | |
|---|---|---|---|
| !Id | Name | City_IsIn.Country | Population |
| Amsterdam | Amsterdam | NL | 770,000 |
| Antwerpen | Antwerpen | BE | 470,000 |
| Barcelona | Barcelona | ES | 1,600,000 |
| Belfast | Belfast | UK | 280,000 |
| Bern | Bern | CH | 127,000 |
| Brussel | Brussel | BE | 1,000,000 |

# Repository (1/2)

The repository is the physical place where the information on the knowledge area is stored.



# Repository (2/2)




The Repository contains:

- .sim files holding a single knowledge element
- .csv files holding multiple knowledge elements

- idem .sim and .csv for the model

- .jpg, .png, files holding figures and diagrams

- .css stylesheet (also: msword template)
- .js javascript glue for the website

- generation options
- output settings
- "views" and "selections" on the information
- other such as .xls, .ppt

The idea is that every output can be reproduced at any time from the repository.

# Engine

The Orde! engine is a Rexx program, 35.000 lines of code.

Evolution: OS/2, NT, 2000, XP, Linux, Regina Rexx.

It runs in a shell and does the following:

- it reads the repository (or part) into memory

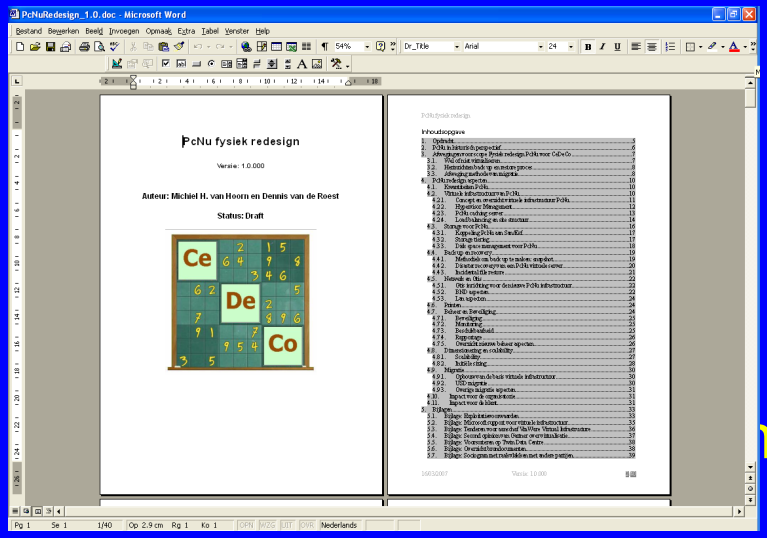
- it solves references by value
- it does cross reference on relations
- it resolves reference by link

- it creates tables
- it generates output

- it exits after generating a document

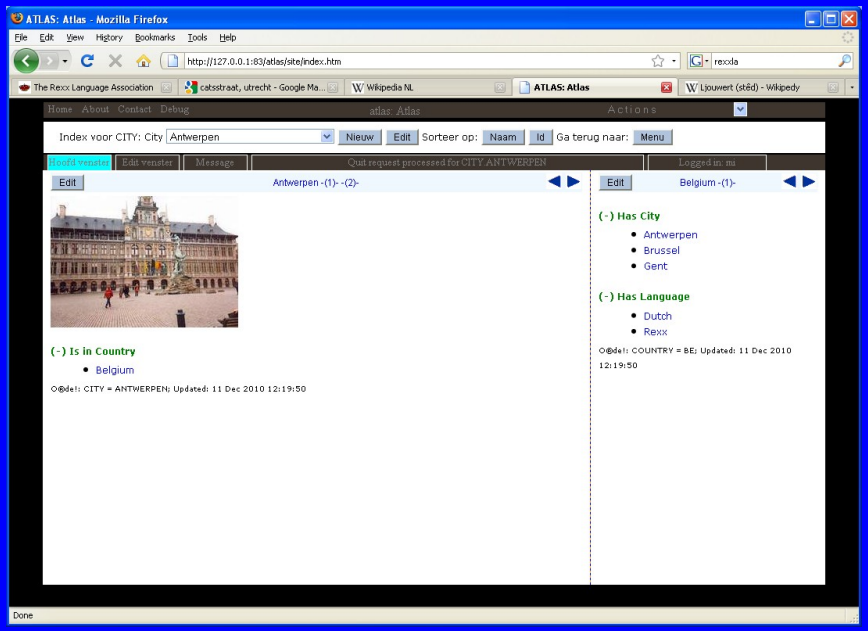
- or it waits for update requests in queue

# Outputs

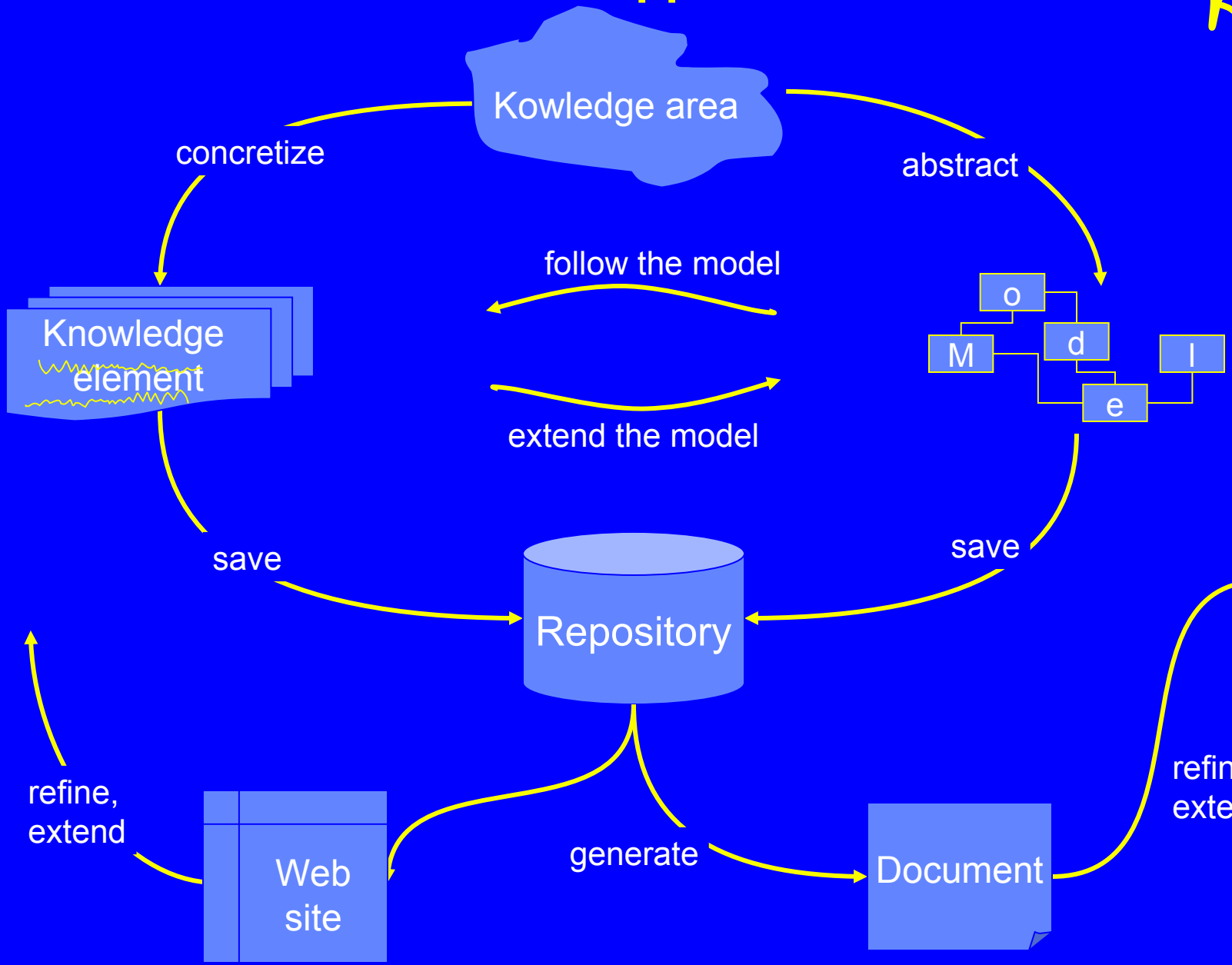



. Html

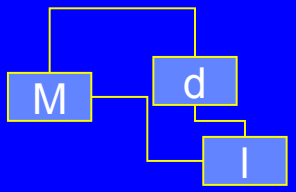
- wiki

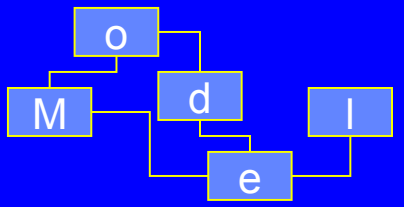


# Orde! approach



# Evolutionairy approach




[Req = …]
Name =…
Has.Sth =…
Descr =…

Starting point

Growth of model

More entities
More relations

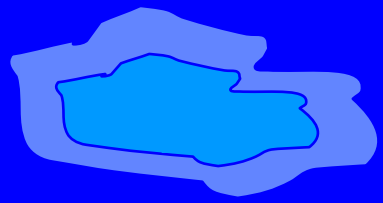
Growth in detailedness

More attributes
Datadictionary

Growth in volume

Larger scope

[Req = …]
Name =…
Has.Sth =…
Descr =…
Satisf =…
RelTo.Sbb =…



# Orde! runtime environment

- Regina Rexx
  . Regina (better performance than OoRexx)

- Any text editor
  . for example X editor

- Browser
  . tested on Firefox, Safari and Internet Explorer

- Javascript
  . for user interface
  . for server based editing

- Optional
  . Web server with CGI for server based editing
  . spreadsheet tool to create .csv
  . tool for capturing hotspot coorinates

and Orde.cmd

# Why use Rexx?

| | |
|---|---|
| Rexx learnt in 1986 | good investment |
| Associative arrays | one of the first |
| Scripted language | readable code |
| Stable over the years | not like MS languages |
| Platform independent | easily portable |
| Good tracing facility | efficient debugging |
| Small runtime | easily installable |

Added for development environment:

- compile program to create single Rexx program
  out of library of subroutines

# How Rexx used? (1/4)

The internal data structure:

three stem variables:

- **struct.** to hold the model
- **data.** to hold the knowledge elements
- **misc.** to hold miscellaneous program data

```
[Country = NL]
Name         =  The Netherlands
Population   = 16.000.000
Description  =
<Name> is a West-European country
on the North Sea. It's capital is
[City.Amsterdam] ...
Country_Has.Lang = Dutch Frysk
```

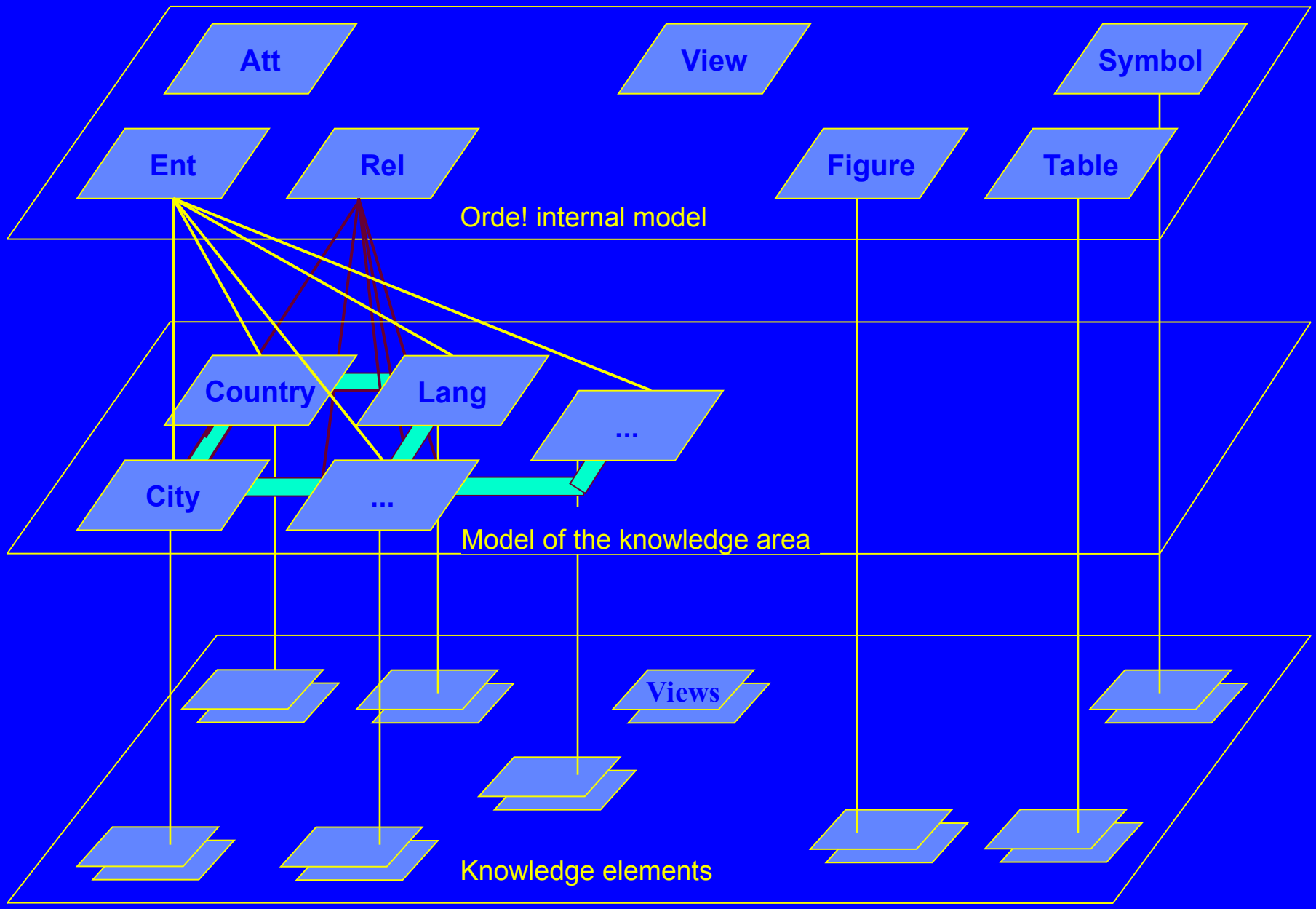
```
struct.8 = "Country"
struct.8.1. = "Name"
struct.8.2 = "Population"
...
```

```
struct.Country = 8
struct.8.Name = 1
struct.8.Population = 2
...
```

```
data.8.1.!Id = NL
data.8.!Nr.NL = 1
data.8.1.Name = "The Netherlands"
data.8.1.Population = 16.000.000
data.8.1.Description = <Name> is in his role as ...
data.8.1.Country_Has.Lang = Dutch Frysk
```

Using this structure, both associative and enumerative access.

# 3-layer structure



# How Rexx used? (2/4)

Exploiting the data structure in parsing

| | |
|---|---|
| Solve by reference | <Name><br><City.Amsterdam;Population> |
| Function | <!Fu.F_Sum; Country :nl esp gr: debt> |
| Solve by link | [City.Amsterdam]<br>[Lang.nl be de; Sep=bullet] |
| Insert figure | [Fig.MapNL] |
| Insert table | [Tbl.City-Tbl]<br>[Tbl.dynamic;*: Tbl_Ent=Country:Tbl_Ids=]] |
| Complex | <Fstp.<Func.<SBB_Has.Func>;Func_Has.Fstp>;<br>Fstp_Uses.Func><br>"All directly or indirectly referenced functions of a SBB" |

# How Rexx used? (3/4)

Subroutine template

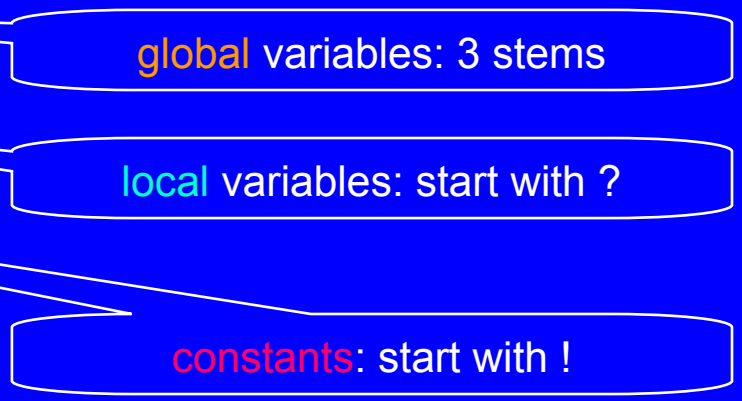
```
ResolveReference:
Procedure Expose struct. data. misc.
Parse Arg ?stn, ?iNr, ?option
?caller = misc.!Routine
misc.!Routine = "ResolveReference"

... code ...

Exit_ResolveReference:

... code ..

misc.!Routine = ?caller
Return .. /* ResolveReference */
```



# How Rexx used? (4/4)

Rexx subfuntions

<!Fu.**function**; **arg1** :**arg2**: ...>

- **Function:** written in Rexx, part of Orde.Cmd
- **argn:** may contain references

<!Fu.**F_Sum;** country : *: debt>

- computes national debt of all countries

<!Fu.**F_Sum;** country : <Lang.EN;Lang_IsIn.Country>: debt>

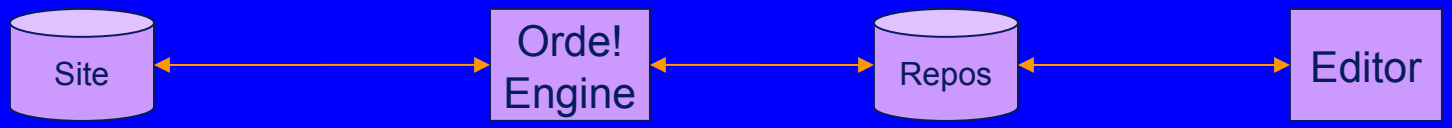
- computes national debt of all english speaking countries

Functions for:

- calculations
- formatting
- dynamic instance creation
- string manipulation
- and ... Interpret

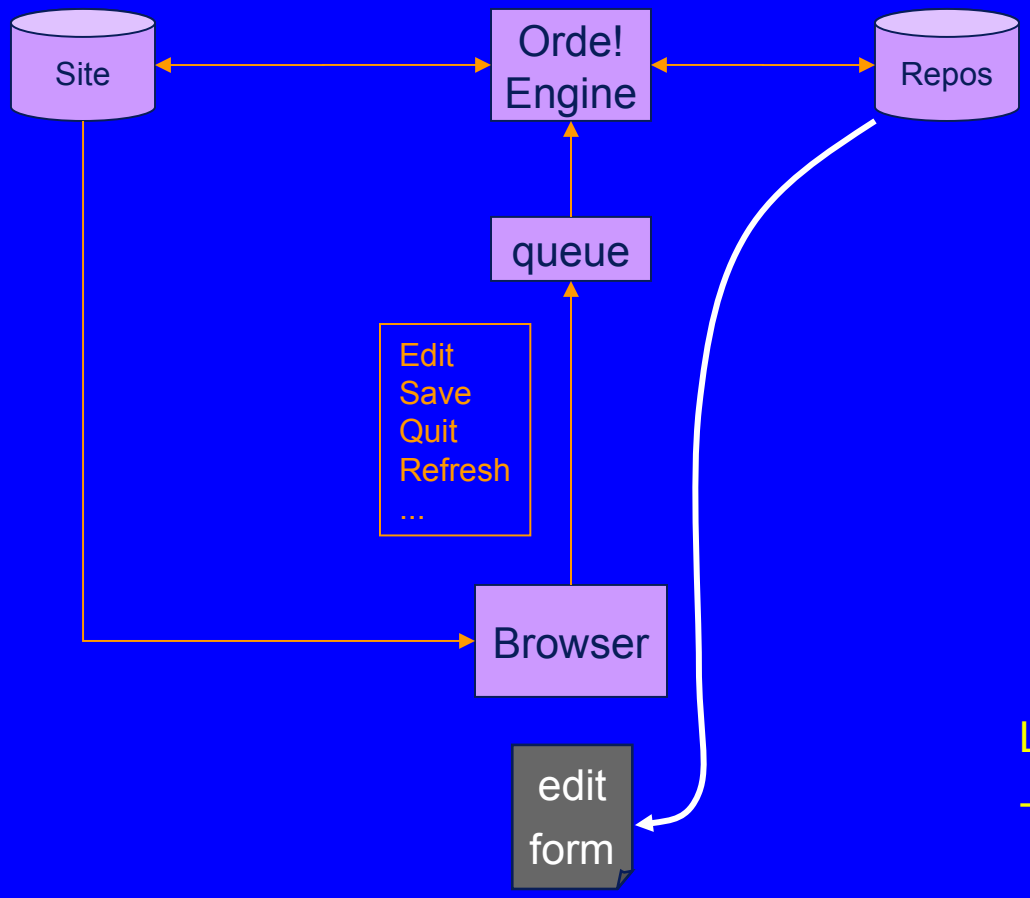
# The editing proces (1/3)

local editing via text editor



# The editing proces (2/3)

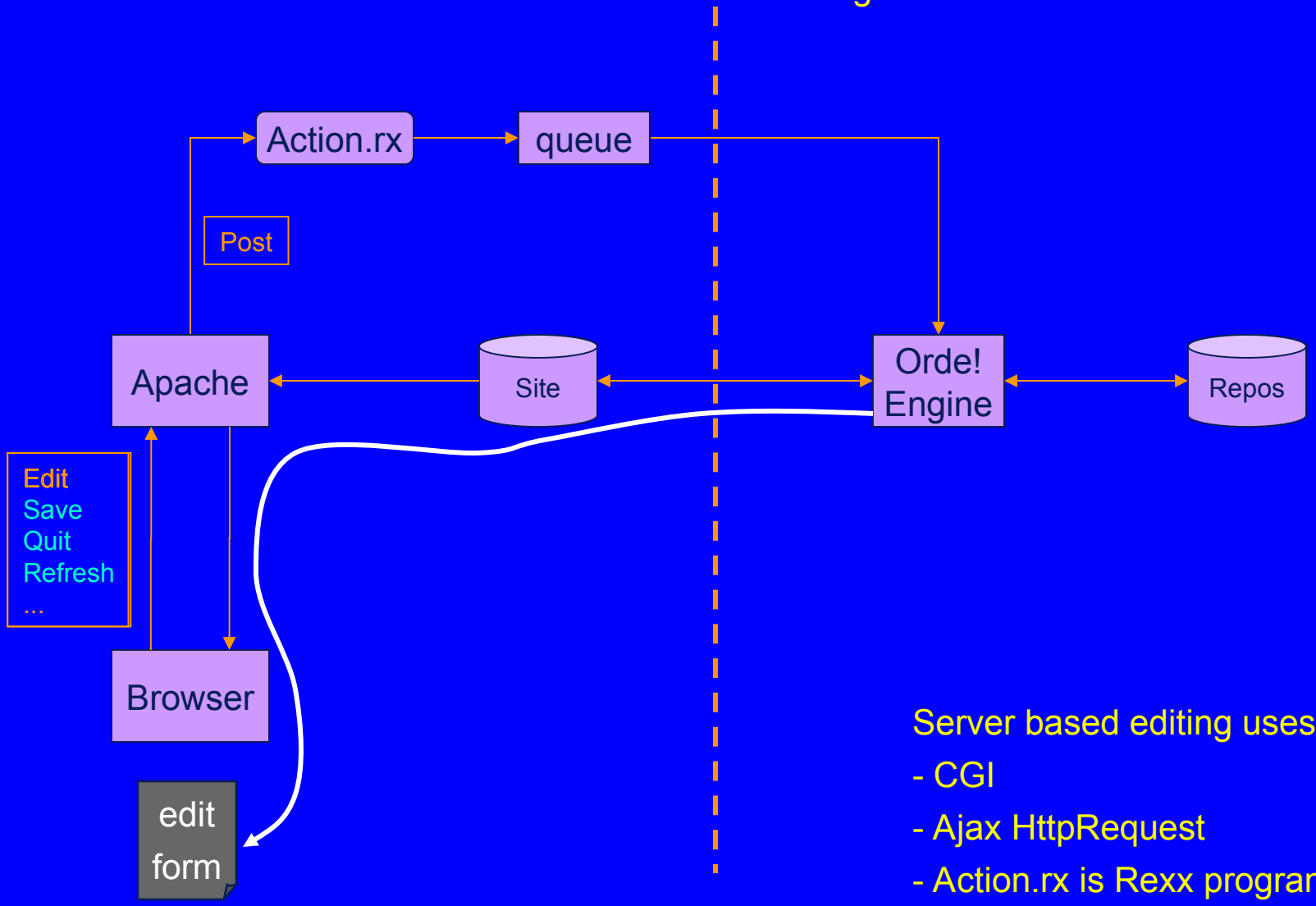
local editing via browser




Local editing uses
- ActiveX to access
  FilesystemObject

# The editing proces (3/3)

server based editing




Server based editing uses

- CGI
- Ajax HttpRequest
- Action.rx is Rexx program

# Applications of Orde!

Knowledge area's
- architectural blue prints
- design documents
- project standards and directives
- organisation model
- data dictionary
- Orde! documentation
- Internet websites

Prototyping
- version control system
- test management system

# Demonstration

# Evolution (1/2)

## Initial features

- Self learning
  - . discovery of entities, attributes and relations
  - . in first phase of project flexibility
  - . in later phases conformance to the model
- Avoid redundancy, improve consistency
  - . store information only once
  - . mechanism of "reference by value"
- Handle figures
  - . create and maintain figures by some tool
  - . mechanism to insert
  - . automatic resizing to predefined formats
- Table facility
  - . easily generate tables from the data in the repository

The generation process ensures reproducing results and using latest versions.

# Evolution (2/2)

Later features

- Input in tabular format via .csv
  - . for bulk input, without much detail
- Multi source input
  - . skeleton structure via .csv
  - . detail via .sim
- Default value mechanism
  - . for example to created compound values
  - . for example, for a naming convention
- Dynamic update website
  - . edit a knowledge element and see the result
  - . updates also dependent knowledge elements
- Transaction layer for prototyping
  - . to add knowledge elements via a web application
  - . in stead of direct editing