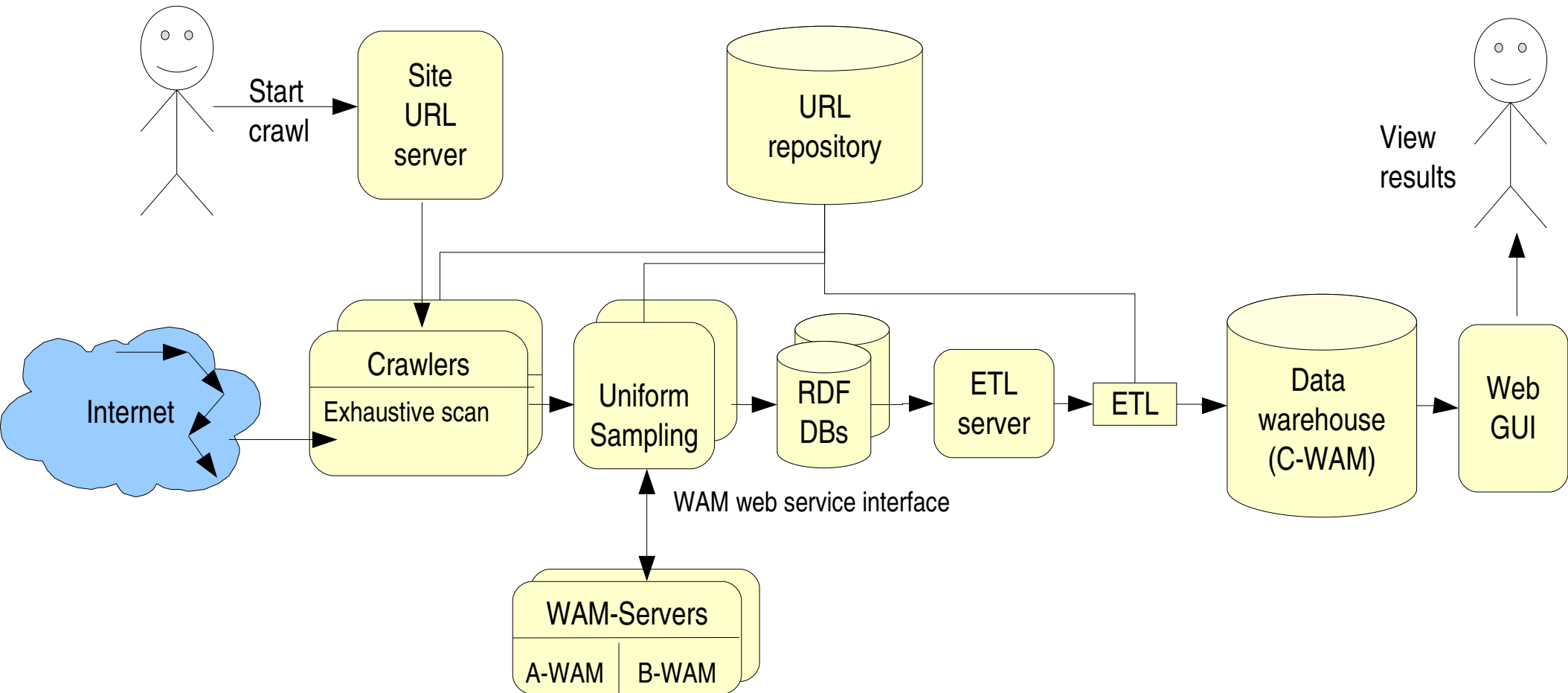


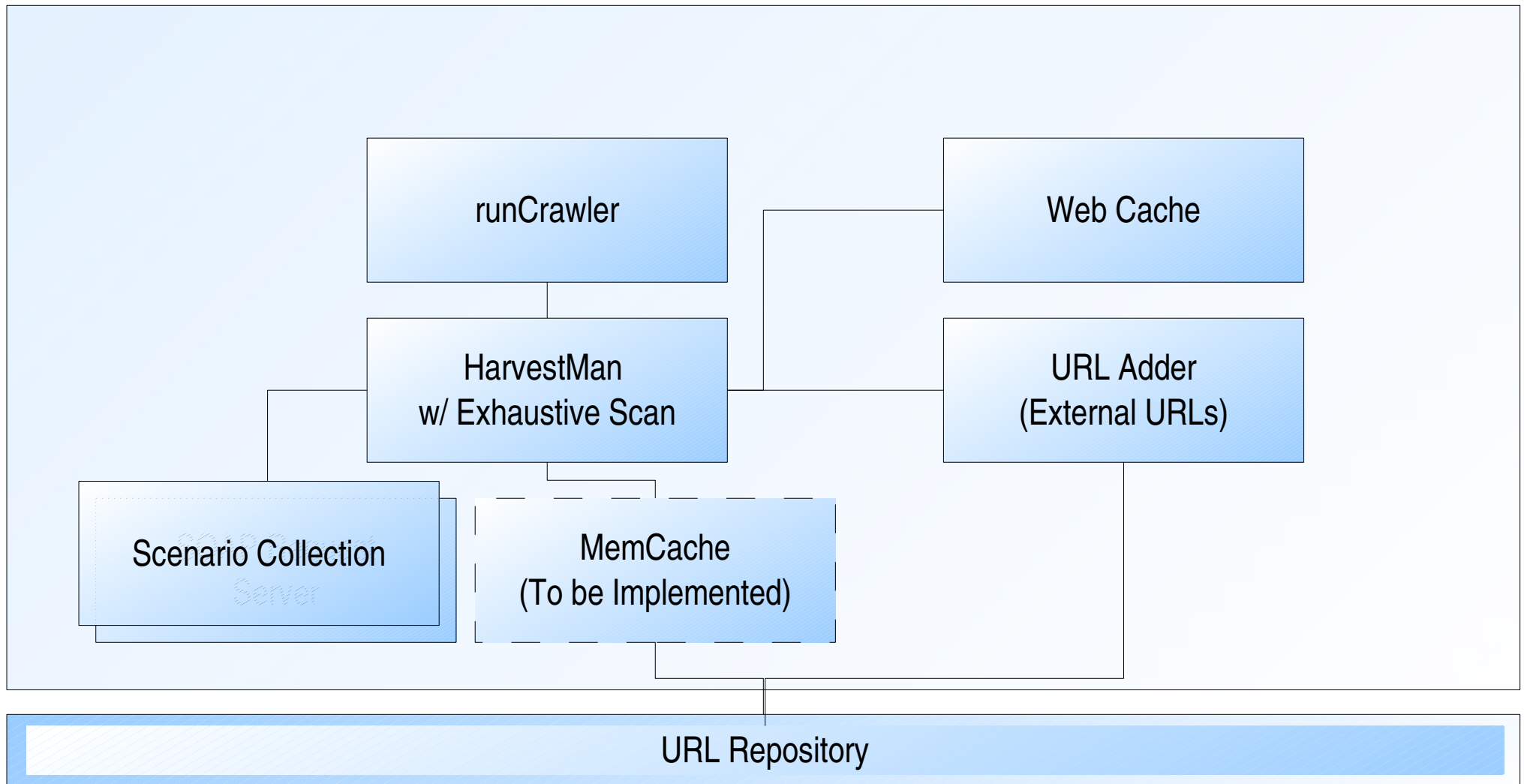
# Architecture Overview

Morten Goodwin Olsen

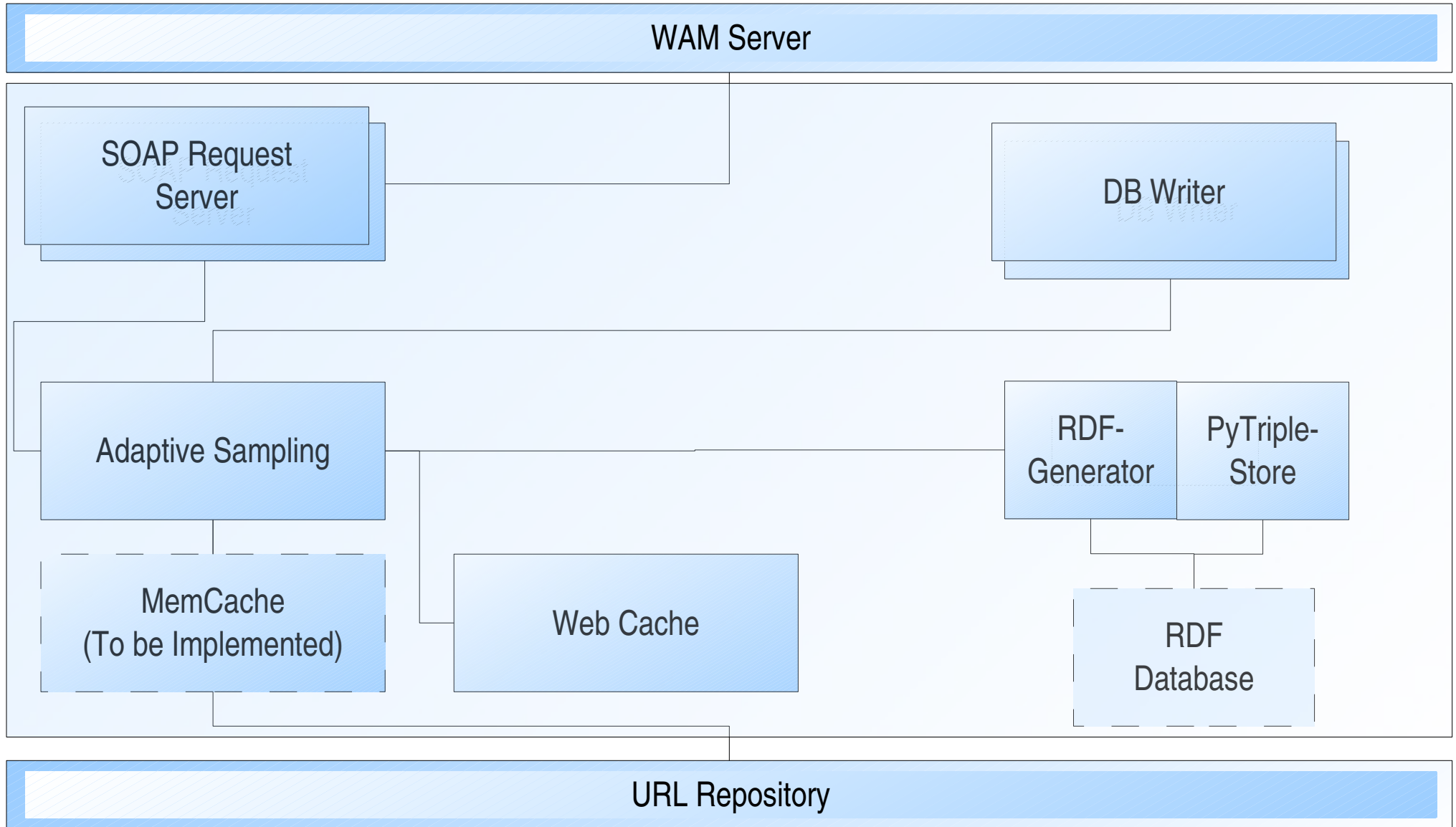
# Architecture Overview



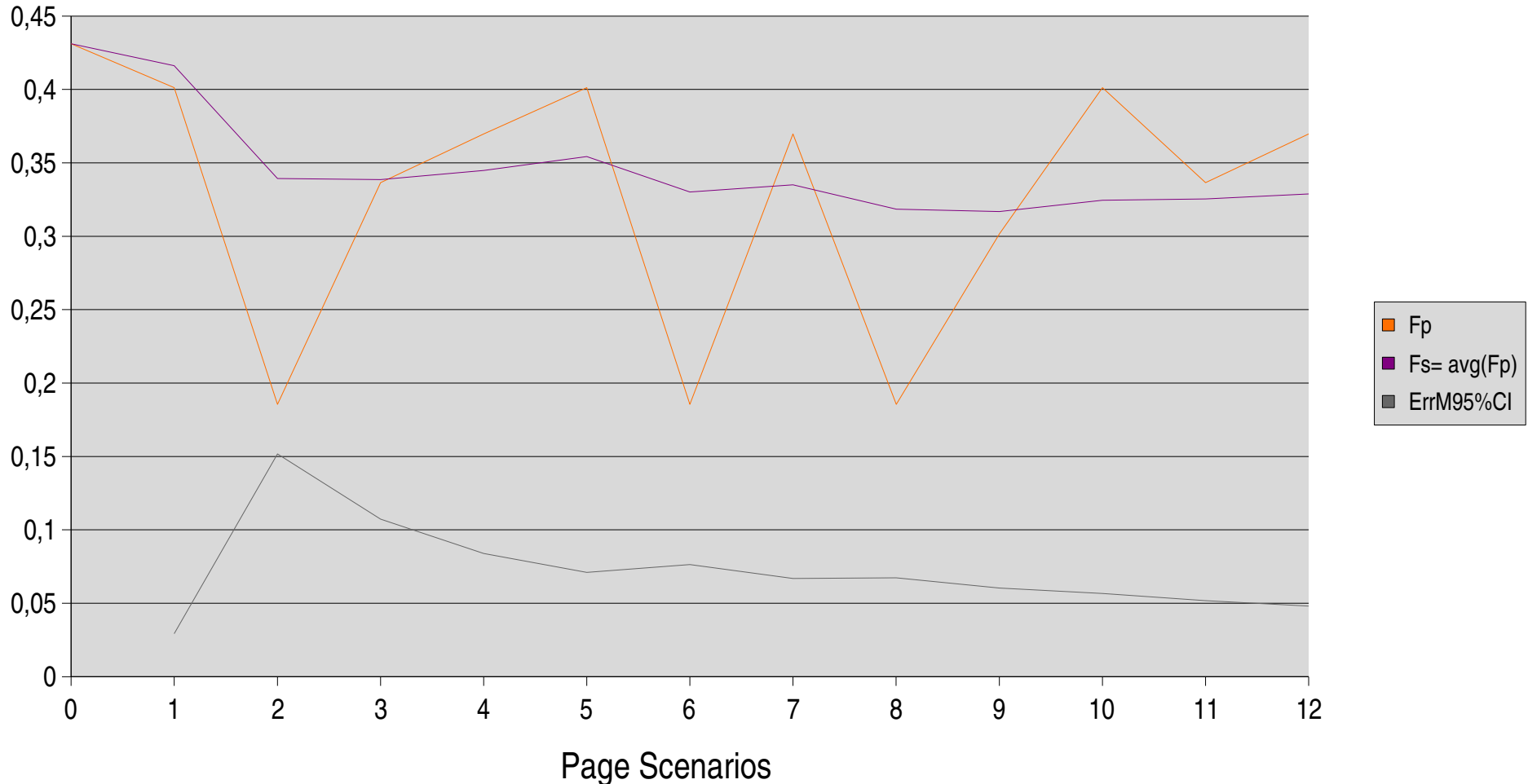
# Crawler – Functionality Overview



# Sampling – Functionality Overview



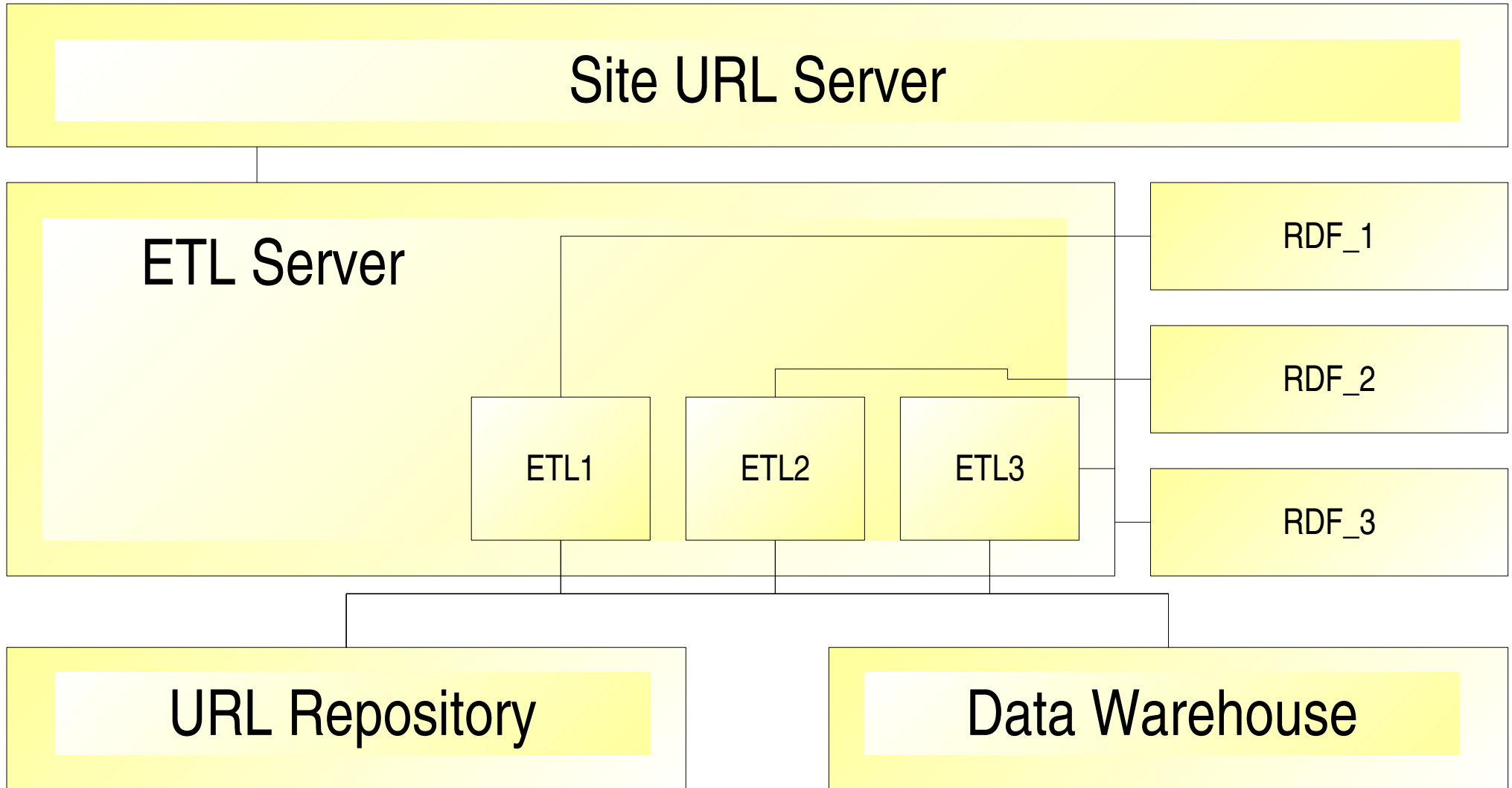
# Sampling Example



- **Example of sampling from a web site.**

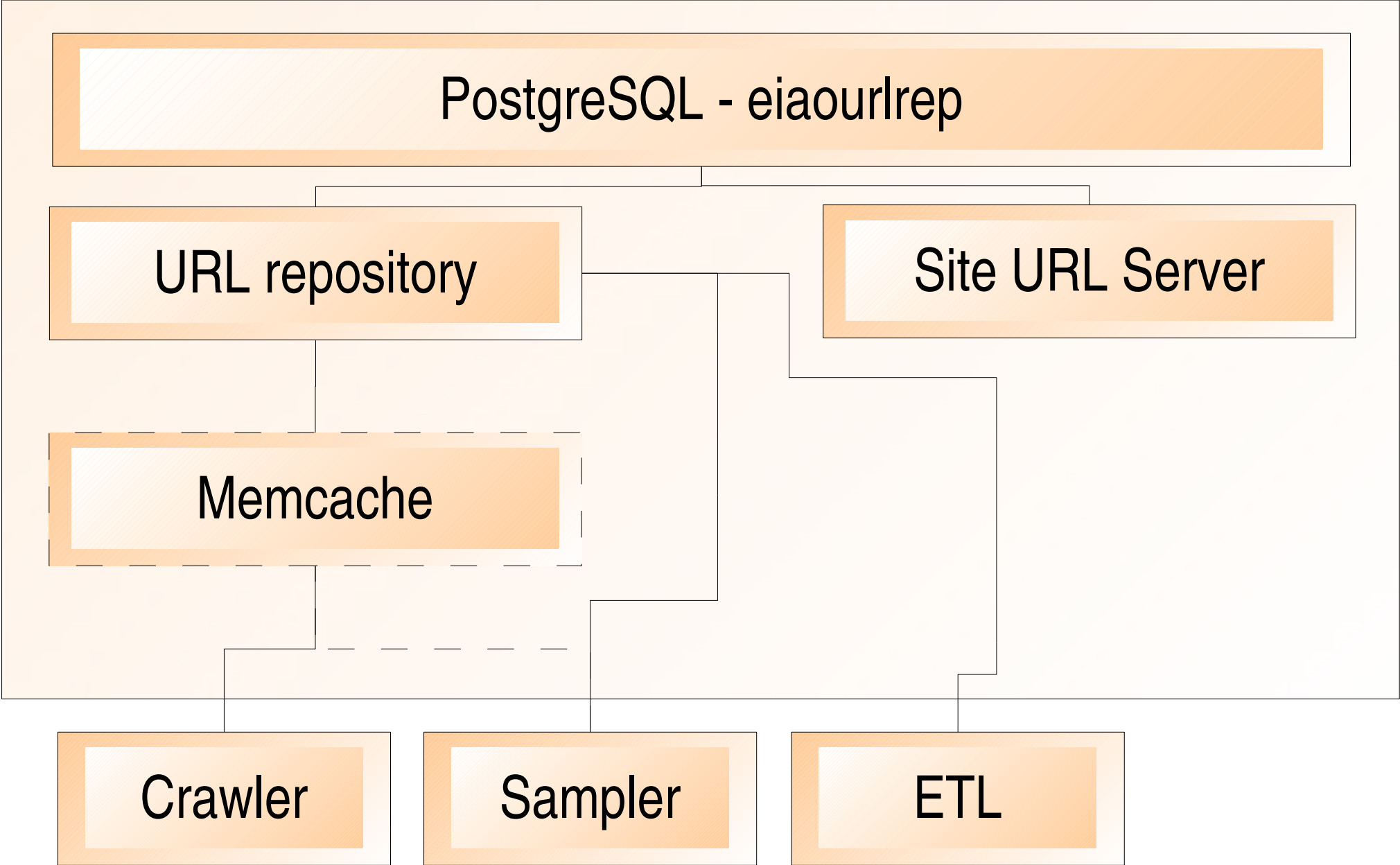
- Note that this example is from the outdated sampling / aggregation. However, the principle is the same.

# ETL Server – Functionality Overview



- Note that the ETLs do not run in complete parallel. However, caching from the RDF repositories can run in parallel.

# URL Repository and Site URL Server



# Watchdog Functionality – Exhaustive Scan



- Crawler sends keepAlive at regular intervals to runCrawler
  - Every 2 minutes
  - I.e. keepAlive (PID,site,host)
- If runCrawler is has not received keepAlive within 5 minutes
  - Kill crawler
  - Restart crawler

# URL Cache Size Problem

- Challenge:
  - Crawling large sites will include a lot of URLs
  - E.g. europa.eu.int
    - Number of URLs according to Google: 2 760 000
    - Memory including wrapper classes etc: BIG
- Solution:
  - When URL not in HarvestMan: Read from memcache
  - When URL not in memcache: Read from Postgres
  - When URL not in Postgres: It has not been downloaded

# URL Cache Size Problem (2)

- How can the crawler know which URLs have been crawled now, and which URLs have been crawled last testrun?
- Solution 1:
  - Keep a testrun stamp at each URL
  - Disadvantage: URL repository will be huge
- Solution 2:
  - Reset URL repository after each testrun
  - Disadvantage: Lose a lot of information that could be useful
- Solution 3:
  - Include current testrun and previous testrun in the URL repository
  - Disadvantage: Lose information that is more than two months old

# Semi-automatic NUTS Categorisation

- 1. Compare Capgemini description with a nuts-list (33/256 - 12%).
- 2. Hostip-lookup (32/256 - 12%)
- 3. Download the main page and perform a lookup of every word with the nuts 3 list (24/256 – 9%)
- Possible extensions:
  - Detect the “About” page to find a description of the location
  - Use <http://geonames.org/>
  - WHOIS if we are allowed
  - Manual categorisation

# Semi-automatic NACE Categorisation

- 1. Classify only using the Gapgemeni description (including very simple stemming, e.g. libraries -> library)
- 2.1 Translate to description to english with the help of babelfish.
- 2.2. lookup the description in wikipedia
- 2.3. See if any of the words written in the first paragraph of wikipedia article is equal to the NACE categories.
  
- Comparing the already provided Gapgemeni description with a NACE list: (118/256 - 46%)
- Automatic Nace categorisation using wikipedia: (35/256 – 13%)

# Semi-automatic NACE Categorisation (2)

- Possible extensions:
  - Manual categorisation
  - Simple pattern recognition based on a smaller sample of manually categorised URLs
  - Base on Yahoo categories? Based on Salient Words  
(<http://www.springerlink.com/content/65plpup3mwv7cnde/>)