LXScript

Scripting with UNIFACE

Gerd Vassen
labsolution

Face to Face Uniface User Conference
Rotterdam, May 2014

labsolution

- founded in 2010, Luxemburg
- 12 employees
- Uniface VAR
- software for the medical laboratory
- standalone modules
- LX
  Laboratory Information System

LIS requirements

- Software to fully support the process-chain in a medical laboratory specimen entry, control and supervision of the measuring process, validation rules, reporting, medical and economical analysis, billing …. to the point of financial accounting, ERP, inventory management
- Automating and process optimisation
- Locally distributed over several sites, international clientele and multilingual UI and reporting (renderZ)
- Hardware environment defined by the customer
- 50.000 new data records … daily (without counting audit/log data)
- High requirements for security and system reliability (no downtime)

LIS interfaces

- Instrument connections: different communication formats and protocols, only partially ‚standardised' (ASTM, HL7, …)
- Hospital information systems (HIS)
- Web order entry and consultation
- Scanner software (order forms, documents)
- Specific external software modules (financial accounting, ERP order entry, result access, other LIS, monitoring and surveillance systems,…)

Interfaces: technical requirements

- Low-Level functions (transport) : TCP/IP, serial, file-transfer (semaphore, ftp, sftp, ...), complex Web-Services, API, XMLRPC, ...
- Realtime-control of the communication processes (multithreading)
- String- and binary data manipulation (checksum, regular expressions, extract and transform, de/encoding, de/encrypting...)
- Automated image processing
- XML/XSLT/XSD (WebServices, reporting server...)
- MS Word, Excel (in/out)

The challenge ….

- Use of a few programming languages that best cover the specific need
- Smooth mutual integration (call-in/call-out, embed/extend)
- Platform independency, scalable, performant, safe investment

Database connection (Oracle), Client/Server, UI, Core-Module

→ Uniface

System-task, customer specific requirements, interfaces

→ ?? Inhouse ? .NET ? Perl ? Lua ? Java ? Python ? Ruby ? TCL ? ...

And the winner is … Python

- modern, large community, well documented
- free, multiplatform (Windows, Linux, Solaris, Mac, …)
- object oriented, clean syntax, simple, exception handling, UTF-8
- extendable through self-written or external modules
- directly deployable (text format), nonetheless performant
- flexible: from the smallest script to complex projects
- ideal as embedded scripting language

LXScript, first part : basics

- learn and understand Python
- first practial experiences on SystemScript-Level (Shell/awk/sed-replacement)
- chose an editor (Eclipse, later PyCharm)
- creation of a framework for faster programming of instrument connection drivers (LX ITF)
- experiment with different integration approaches (embedding/extending Python)
- embed Python via the 3GL Uniface-API Python → LXScript

LXScript, first part : basics

LXScript = extend Python by ufunctions() (Uniface-functions) :

- field access (read, write, $fieldname, ...)
- occurrences (setocc, remocc, discard, $totocc, ...)
- message, putmess, askmess, clrmess (switches → optional parameters)
- global registers ($1-$99 + $$applicationSpecific)
- Activate !   (out1 , out2) = ucall(„Component“, „operation“, in1, in2)
- Other (macro, $applname, debug, $char, commit, ...)

Current state: 42 methods for direct interaction with Uniface

LXScript, first part: basics

More functions for even better interaction with Uniface

listFromUniface: uniface list → python list

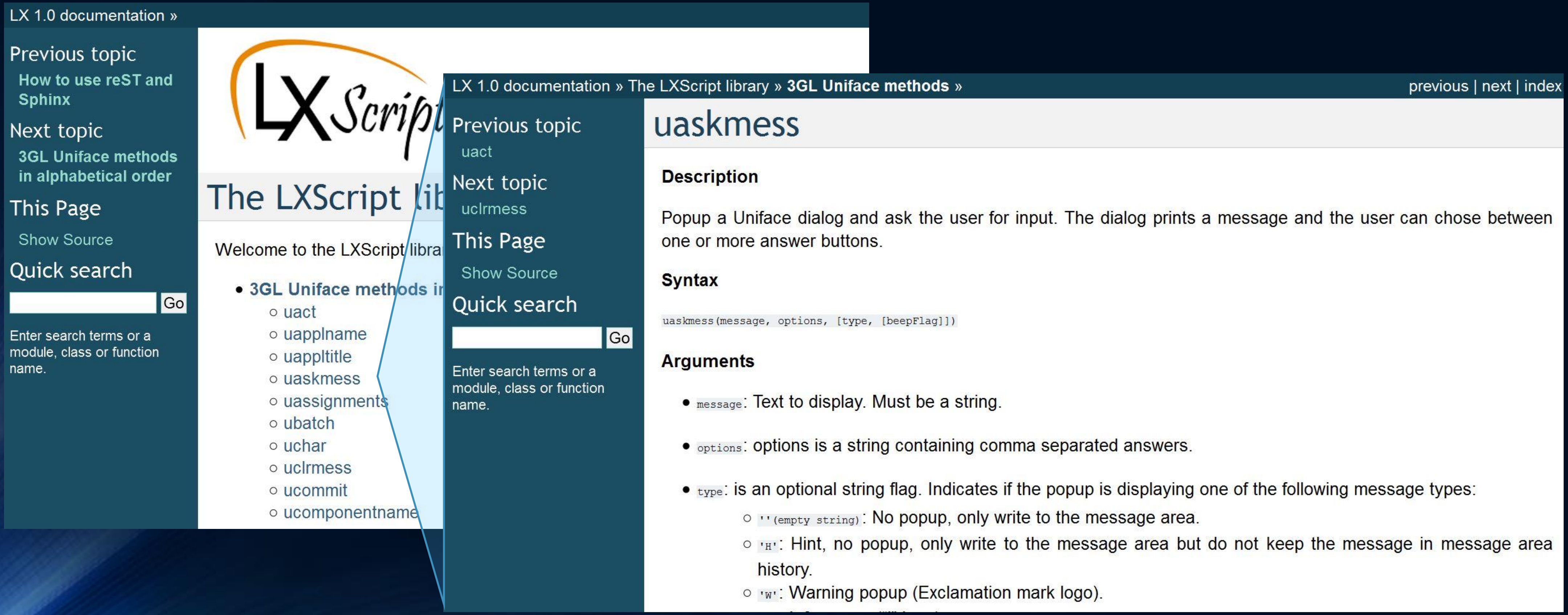
dictToUniface: uniface dict → python hash

toUniface : transform multidimensional python hashes/lists in Uniface dict/list-format (recursively)

→ smooth parameter exchange between Uniface and LXScript

LXScript, first part: basics

Documentation with Sphynx: generate documentation from source code


LX 1.0 documentation »

Previous topic

How to use reST and Sphinx

Next topic

3GL Uniface methods in alphabetical order

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

The LXScript lib

Welcome to the LXScript libra

- 3GL Uniface methods in
  - uact
  - uapplname
  - uappltitle
  - uaskmess
  - uassignments
  - ubatch
  - uchar
  - uclrmess
  - ucommit
  - ucomponentname

LX 1.0 documentation » The LXScript library » **3GL Uniface methods** »

previous | next | index

Previous topic

uact

Next topic

uclrmess

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

uaskmess

Description

Popup a Uniface dialog and ask the user for input. The dialog prints a message and the user can chose between one or more answer buttons.

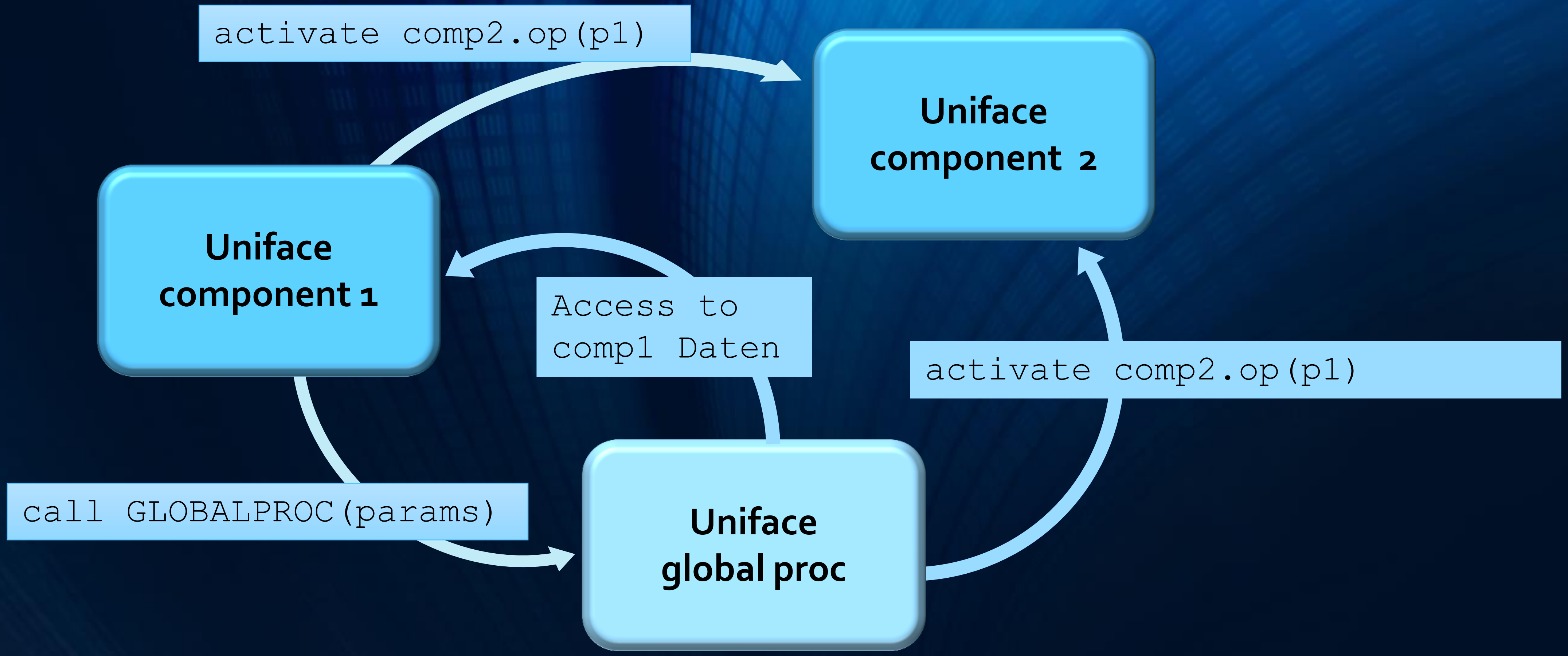
Syntax

```
uaskmess(message, options, [type, [beepFlag]])
```

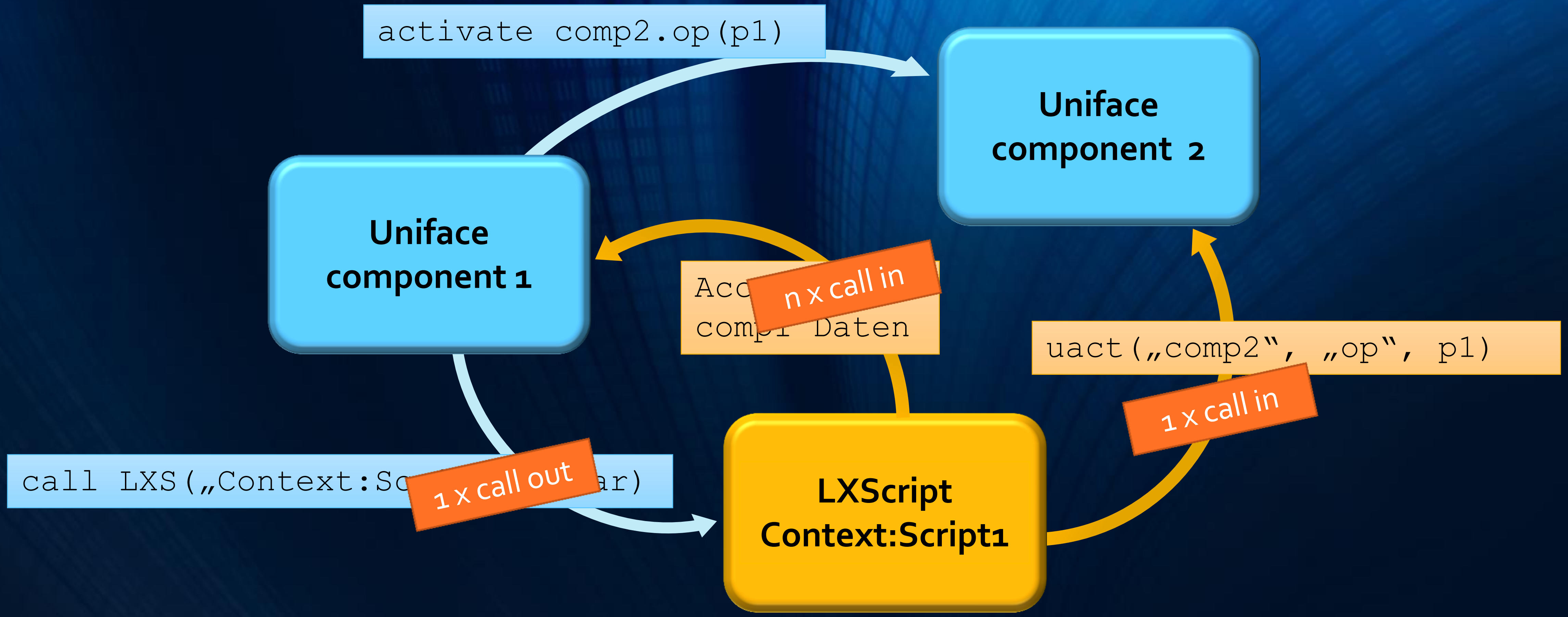
Arguments

- `message`: Text to display. Must be a string.
- `options`: options is a string containing comma separated answers.
- `type`: is an optional string flag. Indicates if the popup is displaying one of the following message types:
  - `''(empty string)`: No popup, only write to the message area.
  - `'H'`: Hint, no popup, only write to the message area but do not keep the message in message area history.
  - `'W'`: Warning popup (Exclamation mark logo).

Uniface Component – global procedure



Interaction Uniface - LXScript



A first example ….

read from and write to fields: *uget, uset*

occurrence manipulation:
positioning, add, delete: *usetocc, ucreocc, udiscocc*

user-interaction message/question: *umsg, uaskmess*

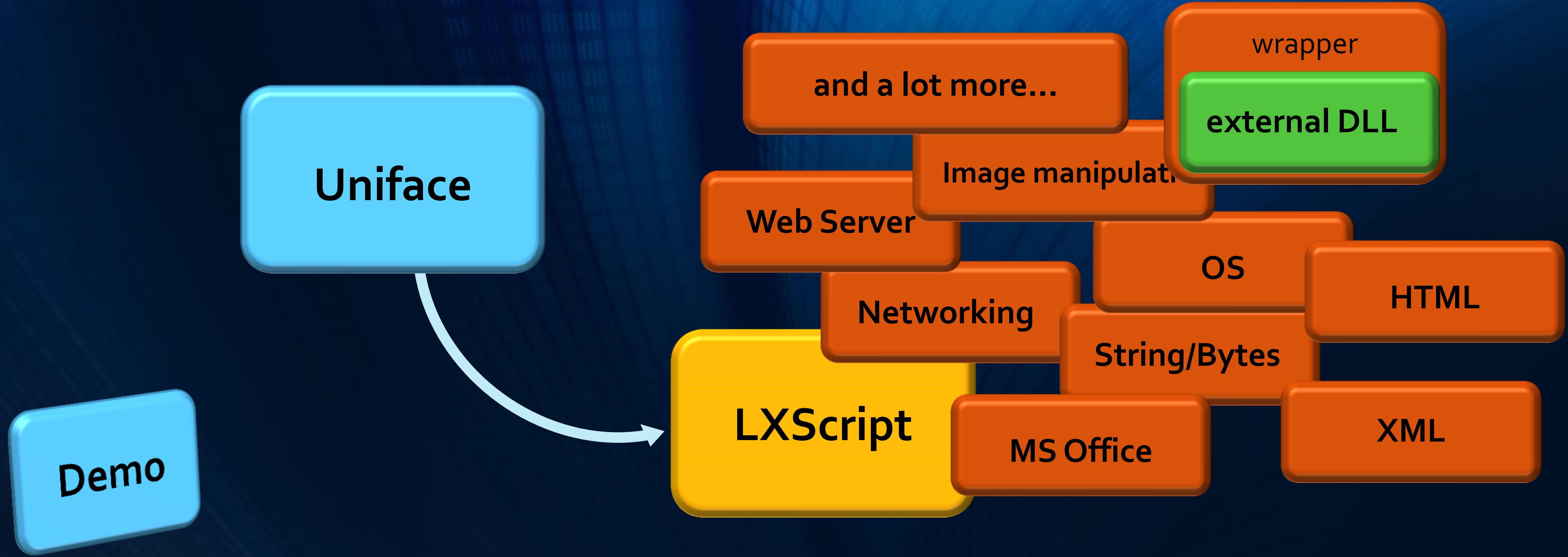
component call: *uactf, umacro*

Demo

Integration von LXScript in LX

- LX Uniface-Part deployment per uar → full version control, for source code and runtime archives (dtap)
- LXScripts as part of the core-application and are deployed the same way, as files
- LXScripts may be customer specific. They are saved in a database table LXScript (similar to other customer data)
- LXScript can call another LXScript → allows to easily vary a standard behaviour, core LXScript → customer LXScript
- Deployment: unit-tests are fully supported LXScript
- Customer can embed his own modules: sales argument!

The door is opened
… without Uniface-extensions !



LXScript, second part : from script to object (OO)

Objectoriented programming with LXScript

- A script can define a class, with proper functions and variables
- An optional context LXContext provides a default-implementation for the class
- LXScript is the inheritance of the LXContext class with full OO-support
- The context also provides documentation and basis-template, which simplifies creation of new LXScripts
- An LXScript is defined by „context:name“ and called the same way

LXScript, Phase 3 : Optimizing the framework integration

Meanwhile the labsolution Uniface-team was not sleeping …

The LX-framework was enriched by a service-component (per Uniface-Template). This _SVC component provides for every entity operations to easily access single or multiple data records of that entity (CRUD).

In the LX-framework all indirect data access passes through those _SVC components

The basic service template has been improved and extended over time (recursive search, SQL-Query integration for best performance, new functions, …)

LXScript, Phase 3 :
Optimizing the framework integration

How to access those SVC service components in LXScript ?

→ new usvc… -wrapper-functions, single line easiest function calls as LXScript counterparts

Instead of uact ‚ENTITY_SVC‘.operation() just a simple usvc …

single occ: getrec, crerec, crurec, updrec, delrec …, getrecfmt, …

multiple occ: getreclist, crereclist … getreclistfmt …

misc: getlabels, getkeyfields, getreccnt …

Demo

LXScript, Phase 4 : JavaScript - Interface

LXScript/JavaScript bridge in the webserver

Functions available in LXScript are made available to the WebServer !

→ LX Cockpit

Monitor and control communication processes

Demo

LXScript, Phase 5 :
LS Data provider (ORM)

We have a OO –language, we want to benefit from that fact and address data structures in an object-like way

Introduce delayed loading in LXScript:

Data structures are filled when they are accessed and only if they are accessed !!

Delayed loading + data model documentation + service wrapper (+ extension methods)

→ new LXScript module ‚LS data provider'

→ ORM object relational mapping of the LX-Database !

LXScript, Phase 5 :
LS data provider (ORM)

How does the LX data provider work ?

LXScript combines

1. Delayed loading (extension load-on-demand)
2. Data model extraced from the Oracle-Data-Modeler (XML analyser)
3. SVC service wrapper (data access via Uniface), Uniface programmed business-logic

→ Data navigation in LXScript via object model

Demo

LXScript

extending Uniface by embedding an extendable language

Gerd Vassen
labsolution

Face to Face Uniface user meeting Rotterdam 15.05.14