URE
living all over me

Stephan Bergmann
Sun Microsystems

URE living all over me

stephan.bergmann@sun.com

1 - **Where URE comes from...**
2 - **...what it has to offer...**
3 - **...and what it lacks**
4 - **Using URE**
5 - **The future**

The History of UNO

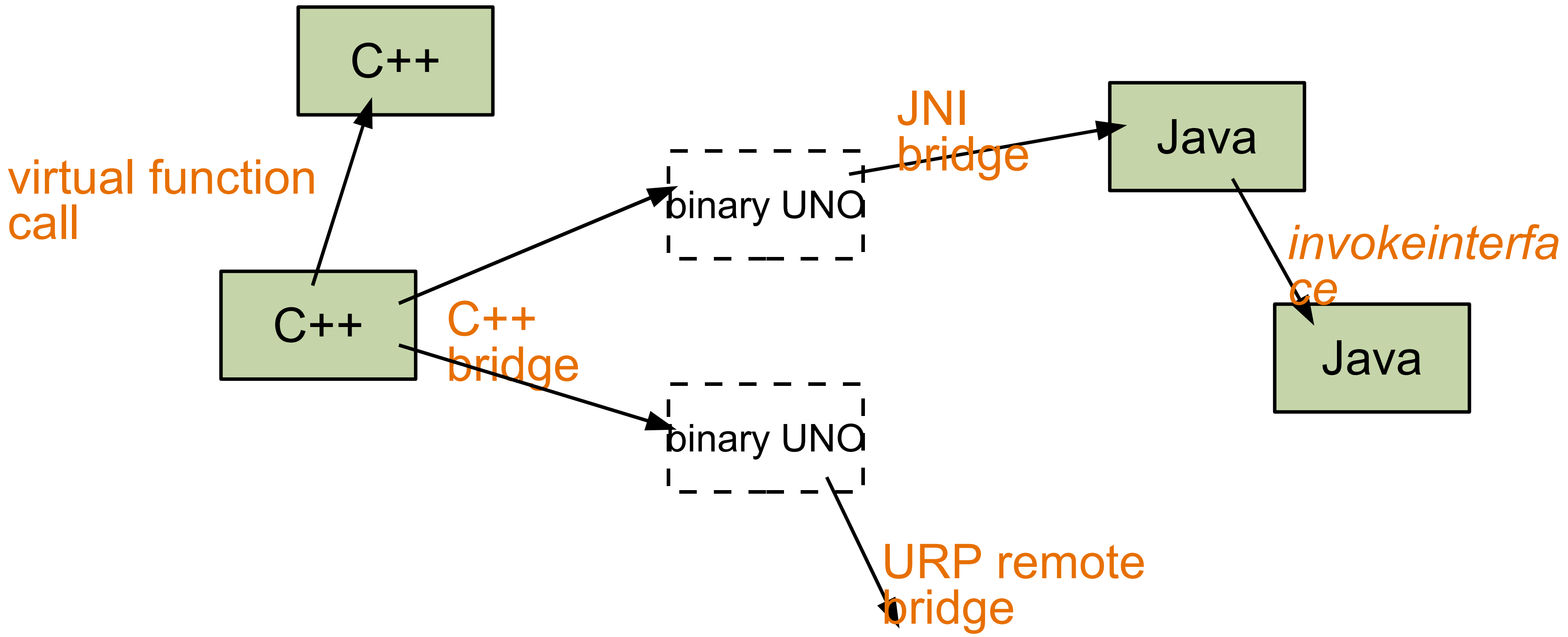
- A component model designed to program OOo:
  - via both add-ons and external applications;
  - in different programming languages (C++, Java, OOo Basic, ...).
- Influenced by Java, COM, Corba, ...
- Some iterations needed to bring it into its current form, “UNO 3.”
- Adventure trips into the world of remote, client–server scenarios (“webtop,” RVP).

UNO Runtime Environment

- By popular (and paying customer's) demand, UNO got extracted from OOo 2.0.
- URE 1.0 is an independent, stable product that will evolve in backward-compatible ways.
- Every large application of today has its own component model. UNO will probably not safe the world. (Or will it?)

What You Get

- Reasonably transparent, reasonably efficient, and type-safe universal method invocation:



What You Get, cont.

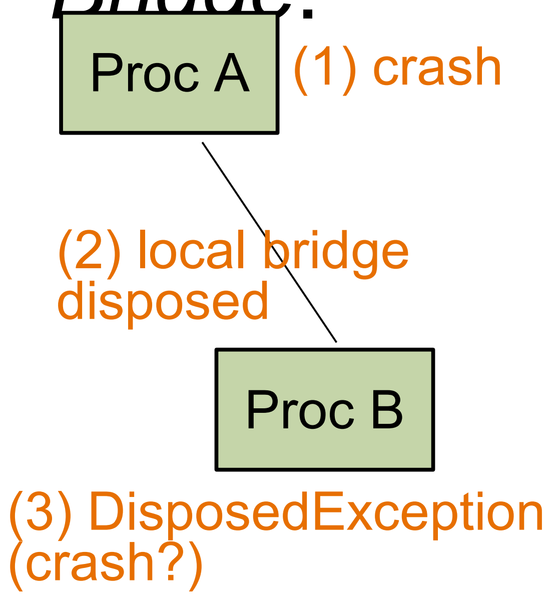
- An infrastructure to build applications from components.
  - > Using the *uno* executable, an application's *XMain* is just a component (and *uno* does all the necessary bootstrapping).
- An infrastructure to add new components with well-defined interfaces to applications.
  - > Extensible/scriptable applications.
  - > Service pools shared by multiple applications (“type-safe dynamic libraries”).

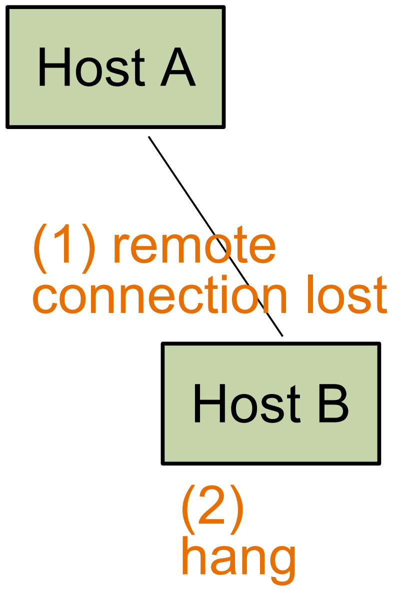
What You Get, cont.

- Little excess baggage (no need to compile in stubs for all the different interfaces and services used).
- Stability: no backward-incompatible changes.
- Cross-platform standards.
- Well-documented:
  - *http://udk.openoffice.org*
  - *Developer's Guide*
  - *dev@udk.openoffice.org*

Pitfalls of Distributed UNO

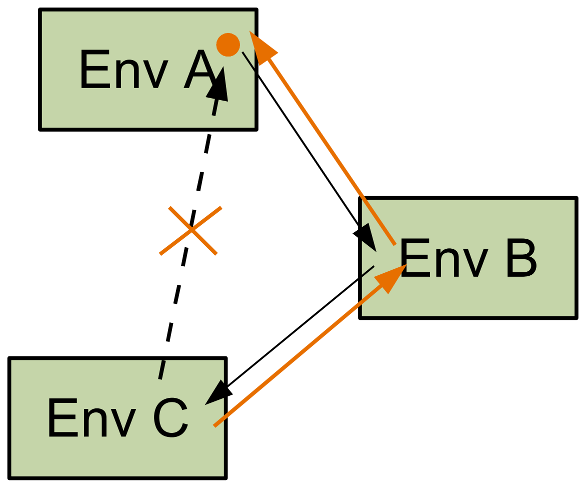
- Tries to be remote transparent. Except for:
  - *DisposedException*;
  - *XComponent/XEventListener* on a *Bridge*.





Pitfalls of Distributed UNO, cont.

- A UNO object is not known by its home, but by its route:



Pitfalls of Distributed UNO, cont.

- In interface design, one size does not fit all:
  - > *sequence<Data> getData();*

    vs.

  - > *interface XDataSequence {*

```
interface XDataSequence {
  long count();
  Data get([in] long index);
};
XDataSequence getData();
```

Security?

- Malicious processes:
  - > At the URP level, access to UNO processes is handled by access rights of the operating system.
- Malicious components:
  - > A native (C++) component cannot reasonably be controlled. (Running it in its own *uno* process is a minimal form of sandboxing.)
  - > In principle, Java components could be controlled by Java's fine-grained security mechanisms.

Practical URE

- Download platform-standard installation sets from *ftp://ftp.stardiv.de/pub/OpenOffice.org/developer/ure_1_0/* (or any other mirror).
- Program it with the OOo SDK:
  - > tools like *idlc*;
  - > UNOIDL files and C++ headers;
  - > *make* environment (optional).
- Tests and examples in *uretest.zip* (CVS module *ure*).

URE on Your Machine

- Well-known home at */opt/openoffice.org/ure*.
- Version dependencies tracked by standard platform mechanisms (*rpm* on Linux).
- Deployment of additional types and services:
  - per machine in */etc/opt/ure*;
  - per user in *~/.ure*;
  - per application.

Future Enhancements: URE

- Component deployment: from *regcomp* to OOo's *unopkg* (which is too big—configuration data, document templates, ...).
- OOo services:
  - > Universal Content Broker?
  - > Configuration?
- Split the SDK and the *Developer's Guide*?
- Include more UNO-enabled languages in the URE (Python!).

Future Enhancements: OOo

- No longer duplicate parts of URE in an OOo installation:
  - reduce size;
  - independent update cycles.
- Versioning/signing UNO components is desirable for OOo. It might be so for other applications as well.
- If UNO gains more widespread use, OOo and other applications can interact with each other more easily.

Where's the cursor?
Where's the eraser?

—Mark E. Smith