Odessa innovation week. Developers day

Interface Definition LanguageS

Ruslan Shevchenko
<ruslan@shevchenko.kiev.ua>
@rssh1

Consultant @ Lynx Capital Partners

Plan

- Historical Overview [1990-2015]
  - from Sun RPC and CGI to Thrift & RESTFull services
- Today-s landscape
  - IDL-s and today application patterns
- Future

Historical Overvies: early times

1990. Web not exists yet.

Xamarin modes is discussed:

- two-sided hyperlinks
- decentralized document servers
- publishing hyperlink = publishing document

Protocols: File exchange.
Terminal Access
Custom

Services on top of command-line

or custom protocols

Historical Overview: ONC RPC

1991 - 1995 Sun (then ONC) RPC

RPC = Remote Procedure Call

Service specs = C structures
+ function definitions

Some RPC-based services are still in use (NFS)

RPC still available in standard Unix-based OS distributives. (Linux/MacOS/FreeBSD)

Historical Overview: ONC RPC (2)

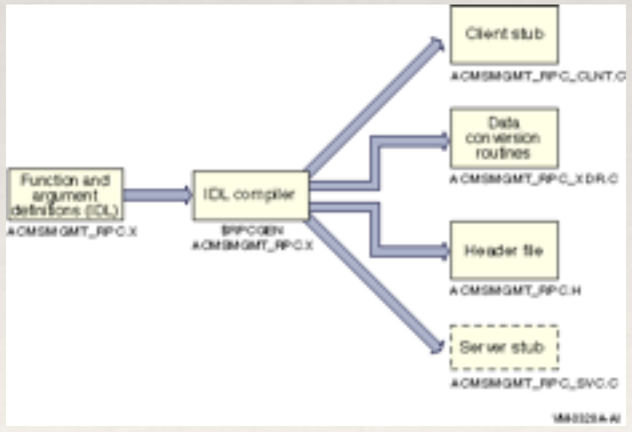
Service specs = C structures
+ function definitions

```
struct request {
    int from;
    int to;
};

struct result {
    int number;
    string line<>;
    struct result *next;
};

typedef result *res_list;
```

```
program DEMO_SERVER {
    version DEMO_VERSION {
        res_list get_line(request) = 1;
    } = 1;
} = 0x20000023;
```



generate code

Server stub

Client stub

Historical Overview: CGI

1993 CGI (NASA http server, than W3C)

CGI = Common Gateway Interface

No service specs.

Inputs of programs are request Name / Value pairs (as strings)

Standard output of external program directed to browser

Historical Overview: CGI (2)

```
local ($buffer, @pairs, $pair, $name, $value, %FORM);
 # Read in text
  # Split information into name/value pairs
 @pairs = split(/&/, $buffer);
 foreach $pair (@pairs)
 {
     ($name, $value) = split(/=/, $pair);
     $value =~ tr/+/ /;
     $value =~ s/%(..)/pack("C", hex($1))/eg;
     $FORM{$name} = $value;
 }
 $first_name = $FORM{first_name};
 $last_name  = $FORM{last_name};
```

```
print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Hello - Second CGI Program</title>";
print "</head>";
print "<body>";
print "&function()";
print "</body>";
print "</html>";

1;
```

Typed/Untyped interfaces

- Typed = IDL-based.
  - RPC
  - CORBA IDL
  - SOAP
  - Zeroc
  - Thrift
  - Google protobuf, gRPC
  - Cap 'n proto
  - Message Pack
  - ……………..

- Untyped …
  - CGI, Fact-CGI, …
  - Plain XML
  - Json, Jsonp
  - RESTfull
  - Auro
  - Hessian
  - Bjson
  - BERT, BERT-RPC
  - ………..

Dichotomy up to today

Historical Overview: CORBA

1991-1993. CORBA-1.0-1.2

- more general architecture guidelines then concrete standard.
- mapping to C

1996. CORBA-2.0

- IIOP (Interoperability)
- IDL mapping to C++ and Smalltalk,

1999 - 2001. Gold Times. CORBA-2.1 - 2.6

- Over 700 participants in OMG.
- W3C want to adopt CORBA as next web API (instead CGI)
- Portable Naming/Event/Service, Realtime specs, Valuetypes
- Official mapping to 14 programming languages (unofficial to alll)

2002. CORBA-3.0

Historical Overview: CORBA

1991-1993. CORBA-1.0-1.2

- more general architecture guidelines then concrete standard.
- mapping to C

1996. CORBA-2.0

- IIOP (Interoperability)
- IDL mapping to C++ and Smalltalk,

1999 - 2001. Gold Times. CORBA-2.1 - 2.6

- Over 700 participants in OMG.
- W3C want to adopt CORBA as next web API (instead CGI)
- Portable Naming/Event/Service, Realtime specs, Valuetypes
- Official mapping to 14 programming languages (unofficial to all)

2002. CORBA-3.0

Nobody cares, Industry have switched to XML

Historical Overview: CORBA

Reasons to migrate away from CORBA to XML-based RPC:

- CORBA Complexity and learning curve.
  - // XML is easy - just print
- XML can be read by human.
- Easy usage from script languages
- XML is HTTP-based / Not need to open firewalls for new ports
- First CORBA systems was too fine-grained and slow.
  - // When you print XML, you native design is coarse-grained

Historical Overview: XML-RPC/SOAP

1998. XML-RPC, (Microsoft)

- more general architecture guidelines then concrete standard.

2000. SOAP-1.1 = Simple Object Access Protocol.

- W3C recommendation
- RPC still guidelines. WSDL-1.0

2002 - 2005. Gold Times. SOAP-1.2 +

- WSDL-1.1 is formalization of WSDL-1.0
- OASIS: more than 100 XMLbased vertical standards.
- UDDI: universal registry of web applications.
- First public draft of WSDL-1.2

2007. WSDL-2.0 (=WSDL-1.2) become W3C standard

Historical Overview: XML-RPC/SOAP

1998. XML-RPC, (Microsoft)

- more general architecture guidelines then concrete standard.

2000. SOAP-1.1 = Simple Object Access Protocol.

- W3C recommendation
- RPC still guidelines. WSDL-1.0

2002 - 2005. Gold Times. SOAP-1.2 +

- WSDL-1.1 is formalization of WSDL-1.0
- OASIS: more than 100 XMLbased vertical standards.
- UDDI: universal registry of web applications.
- First public draft of WSDL-1.2

2007. WSDL-2.0 (=WSDL-1.2) become W3C standard

Nobody cares, Industry have switched to JSON/REST

Historical Overview: XML-RPC/SOAP

Reasons to migrate away from XML-based RPC to JSON:

- WSDL/SOAP Complexity and learning curve.
    - // JSON is easy - just print
- JSON can be read by human.
- Easy usage from script languages.
- XML is too bloated.

Historical Overview: JSON-based API-s

2000. REST = Representational State Transfer.

- Phd of Roy Thomas.
- Architecture guidelines without concrete specification.
- Formulated in terms of XML API.

2001. JSON (Douglas Crockford)

- Used for interaction for client-side javascript

2002. Yahoo implements 1-st public JSON-based web service

2006 - 201(>5). Gold times.

- RESTfull/JSON is 'default' API for web apps
- Validation: JSONPath, JSONScheme, JSONRPC
- Typed extensions [Auro], binary variants [BJSON].

?

Mainstream loop

- Adoption <~~> Complexity
- Sometimes we need to do really complex things.
- Advanced technology become complex
- New loop with 'something simple'.

IDL-based architecture today

Actually useful

- google protocol buffers, grpc: google.
https://developers.google.com/protocol-buffers/

- thrift: facebook.   (opened via submiting to Apache)
https://thrift.apache.org/

- cap-n-proto: Sandstorm.ua
https://capnproto.org/

Pretty close to main ideas of CORBA IDL,
Still have no valuetypes

- Other …
  - zeroc (http://zeroc.com)
  - MessagePack (http://msgpack.org)
  - FlatBuffers, SOE (Simple Object Encoding)
  - ……………………………

IDL-based architecture today

Good as

- Hight-level micro service description.
- Javadoc-like tools (if you steel use Json/REST: look at swagger)
- Static verification of interfaces.

Pay attention too:

- Async calls .
- Message passing interfaces (local API).
- Traceability. (req-id)

IDL-based architecture today

IDL & microservices.

- UI boundaries != IDL boundaries.
- UI = service with interface for:
  - event generation. (req)
  - event consumption
- Discovery (Naming)/ health monitoring
- Design for ~~failure~~ failure analysis
  - Structured logging

Future directions

Learning curve ~ task complexity

Gradual complexity

software development ~ car driving

Thanks for attention.

- Questions, discussions:
    - @rssh1
    - ruslan@shevchenko.kiev.ua
    - https://github.com/rssh