

[Home](#)
[Antennas & propagation](#)
[Broadcast technology](#)
[Cellular telecoms](#)
[Circuits & modules](#)
[Design principles](#)
[Electronic components](#)
[Electronics manufacture](#)
[Formulae & data](#)
[Power management](#)
[RF topics & receivers](#)
[Satellite technology](#)
[Technology history](#)
[Telecoms & networks](#)
[Test & measurement](#)
[Wireless technology](#)
[News](#)
[Bookshop](#)
[Events & exhibitions](#)
[Jobs](#)
[About us](#)
[Privacy Policy](#)
[Advertise with us](#)
[Ads by Google](#)
[Telecom](#)
[Communications](#)
[Network Systems](#)
[Telecommunications](#)

 Navigation: [Home](#) >> [Articles](#) >> This page

18 May 2011

Open Telecommunications Platform, OTP for Open Communications

Francesco Cesarini, Technical Director of Erlang Solutions, discusses the benefits of the Erlang Open Telecom Platform and how it can be used in product development for writing software for telecommunications systems.



Telecommunications technology headlines are dominated these days by new mobile devices and their operating systems. The steady evolutionary march of radio access standards also grabs column inches as too do the latest innovations in OSS/BSS solutions. Middleware developments seem to swing in under the media radar.

In fairness to the telco press, middleware developments can be bewildering and esoteric. Software development is perhaps not viewed as 'telco' so much as pure IT and so it is left to the very specialist press to cover innovations in the field. Middleware though is the unsung hero of telecommunications networks.

Choosing the right middleware can be the difference between an exciting new offering launching on budget and ahead of the rivals or damagingly late and ruinously expensive. With a wide variety of programming languages, such as Erlang, Java, C and C++ available, choosing the right middleware depends on a range of factors.

Erlang Programming Language

Erlang was originally invented by the Ericsson computer science lab as the programming language of choice for the next generation of telecom systems. While Erlang is a powerful programming language used to build distributed, fault tolerant systems with requirements of high availability, these complex systems require middleware in the form of reusable libraries, release, debugging and maintenance tools together with design principles and patterns used to style network architecture.

Ericsson realized this early and initiated a project to address issues in parallel with its first commercial project. Work began on the creation of the Open Telecom Platform, often referred to as OTP. Open, in this instance, stands for the openness of Erlang towards other programming languages, APIs and protocols. While Telecom was chosen when Erlang was only used internally within Ericsson for telecom products, years before it was released as open source. It might have made sense in the mid 90s, but today we say Telecom refers to the distributed, fault tolerant, massively concurrent soft real-time characteristics present in telecom systems, but valid in a wide range of other industry verticals. Platform refers to the use of OTP as middleware in complex systems.

Prior to its release as open source, OTP was used to develop many turnkey telecom solutions with millions of lines of code, including the AXD301 ATM switch, the GGSN IP gateway node and SGSN support node, two components handling core functionality in GPRS networks. This resulted in a well

Recommended Books


[Mobile WiMAX](#)

 Sasan Ahmadi Dr.
[Best Price £42.94](#)
 or Buy New [£57.94](#)

[Privacy Information](#)

[MIMO](#)

 Alain Sibille Dr....
[Best Price £44.69](#)
 or Buy New [£57.94](#)

[Privacy Information](#)

tested code base proven fit for the most demanding soft real-time systems.

But what exactly is OTP?

OTP can be seen as a control system platform for developing, deploying and running Erlang-based systems. Design principles provide software engineering guidelines that enable systems to be developed in a uniform way. Consequently, different programs that do completely different things in the network will have a common structures and functions.

When developing Erlang-based systems, it is not mandatory to use some or any parts of OTP when writing Erlang code. Using it, however, enhances productivity, reduces the overall code base and increases the code quality. It ensures developers do not reinvent the wheel for common design patterns and problems that have already been solved, hiding many of the tricky issues with concurrent and parallel programming.

OTP is made up of three components:

- The Erlang programming language
- A set of libraries, including tools, interfaces and reusable applications
- A set of design principles and patterns describing the system's architecture

OTP's design principles provide software engineering guidelines that enable developers to structure systems in a scalable and fault tolerant way. Different programs that execute completely different tasks will do so using common design patterns. While it is not mandatory to follow these design patterns, it is very advantageous to do so.

OTP building blocks

Java's key components are code threads which are mapped to the OS threads. While the key component when writing an Erlang program is known as a process, they are independent of OS threads. As a result, processes take microseconds to create and use little memory. They can be created and replicated with little overhead, enabling millions to interact concurrently on any system.

In Erlang, the most frequently used process patterns have been implemented in library modules, commonly referred to as OTP behaviours. They contain the generic code framework for concurrency and error handling, simplifying the complexity of concurrent programming and protecting the developer from many common pitfalls.



Reusable architectures using OTP applications

Behaviours are monitored by supervisors, themselves a behaviour, and grouped together in supervision trees. A supervision tree is packaged in an application, creating a building block of an Erlang program.

Ready-made components are packaged as applications. They include databases, management protocol stacks, interfaces towards other languages, monitoring tools; components that can be reused in-between projects.

OTP applications that come with the standard Erlang distribution include the Systems Architecture Support Libraries (SASL), answering common maintenance and operations requirements including packaging, deploying and system upgrade during runtime.

Mnesia, a distributed, soft real-time database management system written in the Erlang programming language, enables transactions across hosts is another very popular application, alongside implementations of protocols and standards such as the simple network management protocol (SNMP), the Common Object Request Broker Architecture (CORBA) or the Interface Definition

Language (IDL). For telecom systems, protocols such as Megaco H248 protocol stack are often complemented with proprietary components such as SIP or Diameter stacks.

A complete Erlang system such as the AXD301 switch or the SGSN and GGSN nodes is a set of loosely coupled applications that interact with each other. Some of these applications have been written by the developer, some are part of the standard Erlang/OTP distribution, and some may be other open source components. They are sequentially loaded and started by a boot script generated from a list of applications and versions. Take care not confuse OTP applications with the more general concept of an application, which usually refers to a more complete system that solves a high-level task.

Erlang OTP's unique feature set provides an ideal platform for developing distributed systems. Implementing similar systems in other languages requires a great deal of effort on features such as inter-node communication, message queues, failure detectors, and client-server abstractions. OTP instead provides a battle-tested implementation of these facilities as part of its standard libraries, making it possible to rapidly prototype systems without significant overhead spent on the fundamental, generic building blocks.

OTP more than a telecoms platform

In December 1998, a few years after the first internal release of OTP, the source code and libraries were released as open source, enabling Erlang to make headway outside of telecoms. New areas of use included banking, financial switches, web services, control systems or messaging frameworks such as instant Messaging and SMS, as they all have very similar characteristics to Telecom applications.

With embedded hardware becoming more powerful, Erlang/OTP is making headways in the embedded space, alongside cloud computing, noSQL databases and multiuser online gaming. These are all systems that have the same reliability and scalability requirements as telecom applications, and as such, benefitted from the features of OTP. OTP also made headways within telecoms outside of Ericsson and is used by companies such as T-Mobile, Nokia, Motorola and AT&T.

Some of the most successful OTP based projects today include EjabberD, an open source XMPP based instant messaging server with an estimated 40 per cent market share of the Jabber market, CouchDB and Riak, to popular databases in the NoSQL movement. Other users of OTP in the cloud computing environment include the Ruby On Rails Hosting companies Engine Yard and Heroku. Github uses Erlang in its infrastructure, as do Amazon and Cloudant to scale their data stores. Nokia's disco project, a map-reduce framework in Erlang provides 90 per cent of Hadoop's functionality with 10 per cent of the code.

Less is more with OTP

There are many benefits to using OTP:

- Less code to develop as a result of the libraries and other components
- Developers have a common programming style and use a component based terminology
- OTP is a solid and well tested code base, so bugs are rare

All of this translates to time to market and lower cost of ownership through lower development and maintainability costs.

The productivity boosts when using OTP are visible in a study done by Ericsson where they estimated that using Erlang/OTP resulted in a four to ten-fold reduction in code compared to using Java or C++. The bug density and line of code productivity, however, remained the same, making them conclude that using Erlang together with OTP gave them a four to ten-fold increase in productivity and quality. These numbers were confirmed in another study at Heriot-Watt University, where rewriting C++ production code resulted in four to twenty fold code reduction. When examining what the individual lines of code, it was obvious that supervisors played a significant role. The defensive programming in Erlang consisted of one per cent of the total code base versus an average of 27 per cent for the C++ applications. Memory management consisted of 11 per cent of the total code base.

While the Erlang programming language is an excellent tool to build massively scalable distributed systems that will never go down, you need more than just a programming language to enable successful implementations. The Open Telecom Platform is the Erlang middleware that provides all of this. Providing interoperability between increasingly fragmented IT systems is about much more than supplying the glue that holds everything together.

About the author

Francesco Cesarini is Technical Director of Erlang Solutions Ltd, a company specialised in high availability, massively concurrent soft real time systems. He has been programming Erlang since 1995 and was on the team who worked on the OTP R1 release. Francesco has worked with start-ups and blue chip companies alike helping them with all the aspects of Erlang based projects – from coding, reviews and architecture designs to setting up development centres. He has taught Erlang to hundreds of developers, testers, support staff and university students. He documented these experiences in Erlang Programming, a book published by O'Reilly Media in 1999. Francesco is also a frequent speaker at conferences worldwide and teaches graduates and undergraduates at the IT University of Gothenburg in Sweden and Oxford University in the UK.



Erlang Solutions specialises in supporting businesses with the creation, integration, delivery and lifetime support of products and services based on the Erlang programming language, from small developers to Fortune 500 corporations. Erlang Solutions is the only company of its kind totally focused on Erlang and the Erlang community, offering industry-leading research, development, training and worldwide support for businesses using Erlang. Erlang Solutions helps its customers to realise the potential of Erlang-based solutions, with all the inherent benefits offered by the shorter time to market, low lifetime cost, extreme reliability and scalability of Erlang. Erlang Solutions has offices in London, Stockholm and Krakow.

Most popular articles in Telecoms & networks

[Open Telecommunications Platform, OTP for Open Communications](#)

[Optical Wireless Communications: An Overview](#)

[Measuring Jitter on Synchronous Ethernet Networks](#)

[Cloud Computing Systems](#)

[Save up to 70% on IT Costs. Cloud Computing Simplified. Call us Today](#)

www.digitalmines.com

[Scanning Software](#)

[Specialists in document scanning Easy to use integrated solutions](#)

www.docdynamix.co.uk

[Unified Comm. Guide](#)

[Understand Unified Communications. 96 page Guide, Plus TCO Analysis.](#)

ShoreTel.com/Unified-Communications



Ads by Google

Radio-Electronics.com is operated and owned by Adrio Communications Ltd and edited by Ian Poole. All information is © Adrio Communications Ltd and may not be copied except for individual personal use. This includes copying material in whatever form into website pages. While every effort is made to ensure the accuracy of the information on Radio-Electronics.com, no liability is accepted for any consequences of using it. By using this site, these terms are accepted. [Privacy Policy](#)
