The ***Community Z Tools*** (CZT) is an effort proposed by Andrew Martin in 2001. It aims to provide open source **interoperable tool support for Standard Z and its extensions**. A series of proposals was made for tools and lines of research, which included not only basic support with parsers and typecheckers, but also more advanced tools like theorem provers, code generators, and refinement calculators. CZT have expanded to cover Z extensions drawing together people committed to its future development and use. It has a set of tools aimed at extensibility and interoperability.

Z being such a mature formal method, what makes new tools interesting? Practically, the number of requests in Z newsgroups and meetings for tool support has risen considerably. Also, it suits the objectives of a UK Grand Challenge in Computer Science Research in producing a set of tools aimed at formal specification and verification (vsr.sourceforge.net/gc6index.htm). Our main objective is threefold: a) provide extensive and reliable tool support for users of Z and its extensions; b) build an open source architecture that is extensible and modular for further tool builders; and c) allow room for extensions to be naturally added. Similarly direction, the RODIN platform (rodin.cs.ncl.ac.uk) is providing methodology, open source architecture, and extensible tool support for Even-B.

The major strengths of CZT are: a) tools interoperability with thorough Z standard conformance using XML; b) modular and extensible object oriented design in Java, with broad and careful use of design patterns and testing methodology to provide a reliable framework for tool builders; c) a wide range of projects (and openness for new projects) allowing a variety of people, from theoretical to practical backgrounds, to contribute and participate. We also focus on capturing the attention of quite important users: students! In this front, we have CZT embedded in favourite development editors, such as Eclipse and jEdit (www.jedit.org), where users can experience the available tools in friendly environments. This should also motivate industry users interested in formal modelling.

The CZT core is formed by an XML exchange format (named ZML) representing Z and its extensions. From XML-schema documents, we automatically generate a series of representation data types, such as Abstract Syntax Trees (AST), and XML-binding documents (JAXB). A thorough architecture of parsers and printers allowing interchange of various formats is available for ZML, *LaTeX*, or UNICODE, with the least effort possible, through automatic code generation and reuse. Apart from supporting Z, there are currently four extensions available that are fairly stable: an object oriented version of Z (**Object-Z**); ***Circus***, a concurrent process algebra combining Z, CSP, and the refinement calculus with Hoare and He's Unifying Theories of Programming (UTP) as the semantics background; timed communicating object Z (**TCOZ**), an integration of object Z and timed CSP also using the UTP; and finally, a pattern language enabling the description of a system of inference rules for Z and its extensions, which is useful in implementing animators and theorem provers. Moreover, section management allowing modular specification for Z and its extensions are also available.

Apart from the CZT core, there are several subprojects under development. Among those there are: the interface projects as plug-ins for Eclipse and jEdit; XML readers and writers for interchange with other tools not in Java; a system of natural deduction inference rules allowing Z schema unfolding, and trivial term rewriting as the basis of a Z theorem prover; an animator for a finite subset of Z (called ZLive), which uses the natural deduction rules; a simple Java-beans interface code generator (called Gaffe), which enables visual animation of Z specifications using ZLive; a model-based testing tool, which allows one to write models for testing purposes, and how to use model-based testing to generate test suites; Z browsing allowing type inspection and cross-reference of well-formed formulae; interfacing with the Z/Eves theorem prover using the Z/Eves XML API; a series of prototype translators between Z and B, standard Z and Spivey's (Z/Eves) Z, ZML and HTML; and so on.

In the IMS in India, there are works in translating Simulink diagrams into **Object-Z**. At the University of York (UK), they are translating Simulink to ***Circus***. The National University in Singapore does **TCOZ** related work. In UFPE Brazil, a refinement calculator for ***Circus*** is being developed using CZT. In York UK, a translator from Circus to Java was built, and a model checker with theorem proving capabilities for ***Circus*** is under development. In Queensland Australia and Sheffield UK, work in translating Z to PVS-SAL for model checking purposes is under way. This synergy shows the diversity of activity, and the broadness of choice within our community. Yet, there are plenty of open proposals still to be filled and new ideas waiting to be discussed! For instance, foreseeable projects includes: a Z to JML translator to allow verification of Java code; interfacing to existing theorem provers, such as ProofPowerZ; further Eclipse plug-ins; formalisation of our architectures; extension of the natural deduction system to include proof tactics; and so forth.

So, why not join in? There are mailing lists for people interested in proposing new projects, taking on going projects, participating in discussions, or just quietly following what is happening until you find something suitable to your interests. For more information, please visit czt.sourceforge.net, where you can freely register on the czt-users and/or czt-devel mailing lists. Alternatively, you could contact us directly via the information below with your thoughts and ideas for contributing to CZT.

Dr. Leo Freitas +44-1904-434753
Department Computer Science, University of York leo@cs.york.ac.uk
Senior Research Associate, HISE Group, Circus Model Checking www.cs.york.ac.uk/~leo