Attila Lendvai

Curriculum Vitae

Name Attila Lendvai
Mobile +36 (70) 316 7947
Email attila@lendvai.name
PGP key 5D5F 45C7 DFCD 0A39
Location Budapest, Hungary
Work permits Hungarian citizenship

(As of Feb 7, 2023; latest version is available online with signature)

Languages

- Fluent **English**
- Native **Hungarian**
- Intermediate **German**
- Basic **Russian**
- Bird's eye view of **Lojban**

Formal Education

Masters in Electrical Engineering, Technical University of Budapest, 1996-2002

Job History

Jul 2001 – Aug 2002 (about 1 year), Budapest
Project Engineer in Professional Services at **Brokat AG**.
I was mainly working in Java, JSP and C++ on the backends and frontends of financial systems. My final thesis at the university was: *An Internet Open Trading Protocol implementation based on EJB and XML messaging*.

Aug 2002 – May 2003 (about 1 year), mostly on the yacht of Charles Simonyi
Joined Charles' **Intentsoft** start-up as a software engineer, later on I became Chief Engineer of the programming team of 5 and Charles' primary contact.
Many aspects of programming are still done today in an informal, and therefore lossy and error-prone way. These non-formalized transformations are basically the phases when a programmer understands and then transforms an informal specification into a running program. In order to minimize these informal transformations, we worked on an IDE that helps to formalize the cooperation between domain experts and programmers. We continued Charles' previous Intentional Programming research.
Better known colleagues besides Charles were Gregor Kiczales and Mik Kersten.

May 2003 – April 2006 (about 3 years),Budapest
Software Engineer at **NETvisor**, a medium sized tech company specialized in networking and software development, mostly for big telco companies and government organizations.

April 2006 – today, Budapest
Co-founder of **M.Wallen Software ltd**.
Together with 3 friends and long-time colleagues we founded a company to give a legal framework to our decade-long cooperation. The company is specialized in developing and operating custom enterprise solutions for big organizations. Noteworthy customers are two Hungarian ministries: the Ministry of Municipalities, and the Ministry of Education.

Strengths and Interests

I perceive myself as an open-minded, critical thinker, both in my private and professional life. It helps in win-win negotiations with people, and is probably also the basis of being quite flexible in switching between following lead and taking the initiative, as needed. Being open-minded is also key in my life-long search for better and better ways of programming computers, as opposed to following the ever-changing hypes of the industry. I found joy in solving problems and puzzles ever since I was a child, and more recently in analyzing and designing complex systems.

Some of my non-professional interests are: economics (Austrian school), monetary systems, cryptos; psychology; communication and linguistics; rational ethics; total human optimization; nature, especially hiking; table tennis; flying helicopters (models and simulation).

Detailed Experience

Childhood Years

My first computer was an Amiga 500, at the age of 12. I got bored of the computer games and my curiosity woke up about how the games actually work, which put down the foundation for my career as a software engineer. Lacking Internet and any expert around me I started digging on my own, and basically through **reverse engineering** games and programs I learned programming in **Motorola 68k assembly language** and learned the basics of the Neumann architecture.

By the age of 16, I was part of the team that wrote the Amiga version of a moderately successful game called Reunion, written in M68k assembly, **programming directly the hardware** of the computer without any operating systems.

University Years

Later on I bought my first PC and did some **Linux programming** in **C**. I also became a **BeOS** beta tester and wrote several programs for it in **C++.** I partially rewrote the BeOS file manager, which was quite successful in the community for its extraordinary speed (due to **multithreading**), for its 100% covered error checking of system calls, and for its detailed error reporting. I started to 'misuse' inferior mainstream languages like C++ based on my knowledge of other, more expressive languages: e.g. in C++ I used macro/template generated inline destructors of stack allocated objects to emulate the missing unwind-protect abstraction, and I used **template meta-programming** (wrote a lib to detach any C++ function call into a separate thread).

At that time the Internet emerged, so I started reading about and experimenting with reflection, metaprogramming, metaobject protocols, orthogonal persistence, dynamic compilation, mutli-stage programming, mostly in the context of Slate and Common Lisp. Then I got involved in the **development of the Slate programming language**, where I implemented various parts of the system.

In my university years I became interested in **cryptography.** Since then I got to know a few practical technologies like PGP and TrueCrypt, and various other, mainly Linux related solutions.

Professional Experience

In my first job at Brokat I've acquired **Java** proficiency in an enterprise environment. With my colleagues we were **teaching seminars in small-groups** for the undergraduate students majoring in IT, focusing on topics around Enterprise Java and other industrial standards.

About a year later a professor of mine introduced me to Charles Simonyi, and I joined them in Charles' Intentsoft start-up. It was a great opportunity to work with historical figures of computing, like Simonyi and Kiczales, and to further refine my bird's eye view on Software Engineering. I've learned the ideas behind **Intentional Programming**, and acquired hands-on experience of programming **Windows** in **C#**.

With Gregor gone, I also left to NETvisor for an offer that better matched my expectations. At NETvisor, around 2004, I created a web based meta-UI using Echo and AspectJ. My framework was capable of rendering a navigable, searchable, editable and customizable presentation of any object-level data stored in an **SQL** database, granted that there was sufficiently detailed meta-data describing its structure. In this case the description was an annotated Hibernate ORM mapping. Once it reached production quality, we started using it to develop and deploy various applications for our clients, some of them still used in 2011. It greatly accelerated initial application development and lowered the costs of the later stages of the software life-cycle.

Entrepreneurship

Having worked together with 3 friends for almost a decade at various organizations, we realized that we will never really have the freedom to choose our tools. We learned all the industry standards, but our frustration only grew by that, and therefore we decided to found our own company to work on the same problems, but with the tools of our choice.

First we implemented an opensource application server in Common Lisp (SBCL). Among other things, it has the following noteworthy features and components written by the three of us:

- a high performance, muti-threaded web server
- an ORM implementation

• a constraint based change propagation lib
• a delimited continuation implementation where continuations can be serialized into the database. It is used in our unique, continuation based Business Process Modeling solution.
• a partial evaluator lib for Common Lisp
• a presentation layer that can render a component based, object oriented description of a user interface to various backends (which includes XHTML with JavaScript, PDF and ODF). Expanding on my previous work in Java, this layer can render a navigable, editable, searchable and customizable presentation of any object-level data if there's also a formal description of the meta-level of the data. Using that it automatically provides about 80-90% of the user interface of typical enterprise projects, and therefore greatly reduces development time and maintenance costs.
• a metamodel library to describe and manage models that in turn describe real-world objects and their relationships. For example if the object level domain is a set of *users* and their *credentials*, then the first meta-level is the common properties of such users, credentials and their possible set of *relationships* (e.g. users and credentials are two classes of *entities* that have a *one-to-many association* between them). The second meta-level in turn is a model that describes the common properties of all such first meta-level models (e.g. *associations* between entities have two or more *association-ends* that point to another entity and have other properties like *cardinality*, *type*, etc). This library helps to manage such tower of models, primarily by providing the necessary API for querying the meta-data. Services like our presentation layer can use that API to display arbitrary object-level data guided by the available meta-data, or to display the first meta-level of an arbitrary model in form of e.g. an entity-relationship diagram.
• a rule based authorization system, that not only enforces the declarative rules, but also can render a human readable document describing the current privileges of the users of the system

We implemented two major projects using this framework. A short technical description of the bigger one follows (around 2007):
• our customer was the Ministry of Municipalities of Hungary, I was the main contact person
• replaced a full-paper process
• very tight deadlines scheduled by the law, e.g. we only had three months between the first meeting with the customer and the system first collecting live data (that was already used in the planning of the 2008 budget of Hungary)
• the system calculated the normative financing of the municipalities ($5b) based on the collected data, and the constantly changing proposals of the budget in the planning phase
• about 5000 users with 500+ parallel at peek times
• run on a **Linux cluster** that was administered by me
• used PostgreSQL for data storage
• every single software component was free and opensource
• it was successfully taken over by another contractor after being operated by us for 2 years

Research Interests

• Fundamentals of New Computing: a new computing system all the way from the metal in 20.000 lines of code; Alan Kay's current focus
• Using Partial Evaluation for Compiler generation (e.g. this paper): Write an interpreter for a highlevel language using only LLVM primitives as the foundational building blocks, and write an interpreter for LLVM itself. Then this interpreter of the highlevel language can either be run in interpreted mode (using the LLVM interpreter), or be partially evaluated to yield a flat list of LLVM instructions, which in turn can be compiled to native CPU instructions and run directly on the hardware. It would make it trivial to switch between the flexibility needed for experimenting with language features (late binding of abstractions), and an optimized mode for performance (early binding of abstractions)).
• Using highlevel languages like Scheme or Common Lisp in embedded environments
• Reflection and metaprogramming (e.g. Reflection and its use)