1 Prolog

Einiges habe ich schon programmiert und in einigen Sprachen.

Begonnen hat alles noch mit Lochkarten und Cobol, dann kam Fortran hinzu und das alles noch zu Zeiten der goldenen Siebziger und Achtziger des letzten Jahrhunderts. Doch während dies immerhin wenigstens Namen sind, die heutzutage noch halbwegs Sinn machen, kam anschließend im Berufsleben eine irgendwie „exotische“ Programmierumgebung hinzu: Im betriebswirtschaftlichen Umfeld gar nicht so unbekannt, vermochte sie ansonsten nur ein erstauntes „He“ hervorzurufen: RPG. Der Erfolg dieser Sprache beruhte dabei weniger auf ihrer Eleganz denn auf der Maschine, auf der sie weit verbreitet war: eine Maschine, die Datenbanken weitgehend automatisiert hatte und solcherart pflegeleicht machte nicht nur hinsichtlich der Hardware, sondern eben auch hinsichtlich der Programmierung.

Dass diese funktionale Sprache zu Zeiten der Objektorientierung keine großen Zukunftsaussichten mehr bot, war vor Jahren schon klar. Also kam zu dem Sprachengewirr noch eine Prise C++ und ein paar vergessene Skriptsprachen hinzu, bis ich auf der ewigen Suche nach der Zukunft auf die 4GL Omnis Studio® stieß. Damit hatte ich eine Sprache gefunden, mit der sehr bequem einerseits Datenbanken integrierbar wurden und sich andererseits herrlich indirekt auf diversen Meta-Ebenen programmieren ließ – das dazu noch richtig flott und effizient.

Genau das brauchte ich, um meine Experimente durchführen zu können. Das Kapital für die Bezahlung vieler Programmierer hatte ich nicht, die Zeit ist auch nicht wirklich dehnbar und trotzdem wollte ich betriebswirtschaftliche Datenbankprogramme erzeugen, die meist nicht gar so klitzeklein sind, wie gerade Kunden sich das so zu denken scheinen. Meine 4GL erlaubte es mir nun, diese ganzen Anforderungen unter einen Hut zu bringen und dabei ein schlankes Backbone für eine Software aus komplexen Komponenten (SOA service oriented architecture) zu erstellen, die auf SQL-Datenbanken aufsetzt. Um dieses Backbone tatsächlich in der Programmierung einsetzen zu können, benötigte ich jedoch noch Analyse-Tools, weil mir schnell klar wurde, dass die indirekte, metadaten-gesteuerte Ereignisverarbeitung von den meisten Leuten

kaum nachvollzogen werden kann – also muss es der Rechner tun. Doch als ich dann zu überlegen begann, wie ich das Ganze auf eine breite Basis stellen könnte, wusste ich auch, dass ich nicht länger mit meiner 4GL arbeiten konnte: Zu wenig Leute kennen das System und das liegt nicht zuletzt an seiner wankelmütigen Preisgestaltung.

Während ich also bis heute begeistert von den Möglichkeiten dieser Sprache bin, machte ich mich trotzdem erneut auf die Suche nach der Programmierung der Zukunft. Dabei stolperte ich über MDA (model driven architecture), Regelbasierung und Frameworks aller Art, wobei der Berg sich höher und höher auftürmte, der zur Einarbeitung notwendig wurde.

Doch wie gesagt – Zeit ist nicht wirklich dehnbar.

So bin ich nun bei Ruby gelandet, einer Sprache, die aus Japan kommend trotzdem dem deutschen Sprichwort folgt „in der Kürze liegt die Würze“, die dabei Meta-Programmierung geradezu herauszufordern verspricht – und darüber hinaus bereits recht verbreitet ist. Sie versprach also die Vorzüge meiner bisherigen Sprache ohne deren Nachteile zu besitzen.

Genau dies will ich nun überprüfen – indem ich ein kleines, praktisches Programm aus der einen in die andere Umgebung übertrage und dabei protokolliere, was geht und was nicht geht.

Und wie einfach es wirklich geht.

Und wie schnell es zu erlernen ist.