The Genesis of GeneXus

from a Scrappy Tech Startup in South America to a Global AI Enterprise Software Company

by Breogán Gonda | Nicolás Jodal

Introduction

We have been asked on numerous occasions—both from within and from outside the GeneXus Community—to write the story of GeneXus, its origins, and our inspiration for launching the company 30 years ago.

Nicolás Jodal, Chief Executive Officer

In our opinion, telling the story of how we came up with GeneXus wouldn't be an interesting story, in and of itself, beyond highlighting the strong determination combined with endless faith needed to get off the ground (like all entrepreneurs).

Breogán Gonda, Chairman of the Board

More important than that is that GeneXus is, in fact, the product of a great team of people with a high degree of scientific and technological qualifications, enthusiastically working together, and always enjoying what they do, even when facing difficulties that are inherent in launching a tech company whose bold ideas challenge "business as usual."

Moreover, we must also mention that GeneXus as we know it today, and most of all tomorrow's GeneXus, would not be a reality without the GeneXus Community— over 130,000 developers worldwide who create software solutions using GeneXus.

What led us to launch GeneXus?

Before GeneXus, we started out consulting and teaching others about databases.

While this was great for our clients who liked and benefited greatly from our services, it wasn't so great for us. Most frequently, clients called us for help, not when they were defining a project, but when problems arose long after the launch of a project (problems like loss of database integrity, too lengthy response times, and so on).

In those days, companies implemented several databases, each one for treating a specific type of problem. *Corporate databases did not exist*.

When measured in their millions of records, those databases were considerably large, but they were quite small from the viewpoint of information completeness, and specifically in relation to their usefulness for providing support to the company for decision-making.

Each client's databases were quite redundant and, consequently, inconsistent. Such lack of consistency made the combination of using data from the different databases problematic, even when originating in the same company.

Some of the critical questions that arose in such circumstances were:

- Are databases actually useful?
- Is our activity as consultants in fact helpful?

1984: the Year That an Incidental Event Led to the Launch of GeneXus

Sometimes unexpected challenges lead to the greatest discoveries.

The year was 1984, and we working on a significant consulting project that eventually changed our lives, and how thousands of companies would work, forever.

Our client was a big sports apparel and footwear company based in São Paulo, Brazil. The firm's General Manager, our point of contact in the company, was absolutely clear in his thinking:

> *"In this firm, all employees at the intermediate level, those whose decisions are not significant, always have available a great degree of data support, while I, as well as other personnel in the higher management level, can never rely on the proper information to make our decisions, and our decisions are those that could make the difference between the company's success or its failure.*
>
> *It's useless to try to define in advance the information that I will need to make my decisions. Every case is different from the others, and the information necessary can only be defined on the spot, when we actually need it: it is vital for us to define it ourselves and to obtain it immediately.*
>
> *I have realized that we need all our systems to resort to one single corporate database that will allow us to obtain whatever data we need from it, anytime. We need you to completely re-engineer our computer systems in order to achieve this!"*

Clearly, we were facing a huge challenge, as well as a great opportunity. By then, corporate databases and corporate information were common, mainly in developed countries, though not much was being done in that respect. When having to face a large-scale problem, like this client, where no prior, similar experiences have been recorded, risks become a significant issue.

But this was the chance we had been dreaming of—the opportunity to create a solution that could really make a difference for large businesses. We were eager about (and appreciative of) the magnitude of the opportunity we had been given. More importantly, we felt totally confident that we could deliver above and beyond what was expected. We accepted our client's offer to tackle this unique problem.

In the primary data analysis of the project, we identified approximately 100 entities, and estimated that the full model would include over 500 tables (the actual model finally had 750 tables). This was a situation quite different from the usual models in companies at that time, which never implied more than 40 tables.

<u>First Problem</u>: ***How could we enable end users to formulate the queries they needed to make?***

Supposing we managed to build the database and the programs necessary for the application, how could we enable end users or their assistants to formulate queries as they became necessary?

This was in fact a new problem. The ideal thing would be to solve it with SQL, but then we immediately realized this was unrealistic. From the usability point of view, SQL is a low-level language, since it calls for in-depth knowledge of the database.

We then identified the next non-traditional problem to be solved: ***How do you interrogate the database at anytime, with queries that are not foreseeable, and defined by non-technical staff?***

SQL was not the solution, and neither were the “user-oriented languages” available at the time. These were user-friendly but in no way helped to solve the problem of large models.

The first goal of our research was to obtain a language whereby the system would be responsible for selecting all tables necessary, and for navigating through them, all of which should be done <u>*automatically*</u>.

Data Analysis. We started with the usual detailed data analysis. Until then, we had worked with the Entity Relationship (ER) model that dated back to the 60s, when it was introduced by Charles Bachman and later made popular by Peter Chen. We searched the organization for objects relevant to the problem and their relationships, and then represented them in the ER model. But in this case, in principle, everything was relevant since the objective was to obtain a corporate database.

We soon realized that this procedure, quite useful in small-sized models, presented several problems when applied to large-scale corporate models.

Also, we realized the usefulness of visualizing parts of the model with an ER chart. That is: the ER model seemed to be desirable output, but was a useless input in real corporate models.

<u>Second Problem</u>: ***How can we build and manage large models?***

Here we had data models much larger than those we were used to, and we were certainly bound to incurring numerous technical errors derived from the large scale involved!

To temporarily help with this problem, we started by implementing small tools, while we sought tools from international scenarios that could be of help to us. But unfortunately, we found none.

Other problems. The difficulties derived from the model's size soon proved to be just the "tip of the iceberg."

Third Problem: ***Where is the knowledge?***

Who, in the entire organization, would know the data with the necessary degree of objectivity and detail? The reply was categorical: NO ONE!

So, what could we do? Can we solve a problem if we don't have the proper knowledge for it?: No, we can't.

Are there possible palliative measures like training users or scattering "section data managers" throughout the company?

Yes, that is always possible, but when we suddenly encounter problems that are 10 times larger than usual, those palliative measures are hardly sufficient. Clearly, this was a time for us to rethink everything with a greater degree of freedom and a realization that background information and bibliography would be minimally helpful, if helpful at all.

Where is valid knowledge located? Is it possible for us to substitute data knowledge, which we have proven to be non-existent, with other objective knowledge with the detail necessary to allow an inferred data model from it?

The Artist's Vision

When we take a brief look at early art history, particularly drawing and painting, we can see clearly the great differences between art then, and artwork today.

What were man's drawings like at first? How did we draw based on intuition? What are children's drawings like? All answers implied having knowledge of the object to be drawn, and better yet, the possibility of touching it, and getting to know its nature and all possible details. Then we would draw things **"as we knew them."**

And what are primitive drawings like? They are significantly distorted.

Filippo Brunelleschi

And then one day, the technique of "perspective" was conquered. During the Renaissance, certain artists—as well as architects who encountered even more difficulties than artists in producing their drawings—started considering a change of the paradigm from **"we draw things as we know them,"** to **"we draw things as we *see* them."**

In Florence, in 1417, in order to produce perspective drawings of buildings, the Italian artist and architect, Filippo Brunelleschi, came up with a set of rules (principles of descriptive geometry) that we still use today.

Like with every new paradigm, there was initially a lot of resistance to perspective drawing (art critics insisted that "the art of drawing and painting would be replaced by a prosaic and totally uncreative technique"). However, perspective had come to stay and, in fact, it fully displaced all previous techniques.

When a decision is made to rethink everything and the thinking is done with absolute freedom, away from all possible pre-established contexts, nothing is to be discarded in advance. The story of perspective drawing gave us a great source of inspiration.

The most important revelation was the basic principle of changing paradigms: the change was from a complex, confusing and subjective approach, to another one that proved descriptive, simple and objective. *If only we could do something similar with data! Then we would reach our objective even quicker.*

Now, all things considered, we should go back to the question: "***Who, in the organization, would know the data with the necessary degree of objectivity and detail?***" to try and replace it with another question that could be answered with a YES, which would help us in building the data model that we needed.

What are the topics where we have objective and detailed knowledge?

In our search, we found out that, just as there is lack of a good knowledge on data, every user has indeed a very good knowledge of the visions of those data she resorts to.

Going back to the issue of perspective, we could say that: both in perspective drawing and in data analysis, what we do is "to describe visions."

As promising as it seems, in perspective drawing we draw our vision directly, while here, beyond the significance of visions, what we search for is the data model.

The obvious question then is: ***Having a number of user visions, can we infer a satisfactory data model from them?***

This is a very good question because it takes the problem to the sphere of mathematics. First of all, we must define a reference framework.

Our data elements (attributes) will be identified by name and they will basically abide by certain simple rules:

- One attribute will always be identified under a single name, regardless of where it is within the model.
- There will not be any two different attributes with the same name.
- We shall assign names so that they best represent the meaning of each attribute.

In second place, we must represent the structure of visions. Every vision will have one or several attributes organized according to a specific structure.

However, it is never a good idea to reinvent the wheel.

By now, we find that mainly Jean Dominique Warnier and our friend Ken Orr on one side, and also Michael Jackson on the other, had all made great advances in describing data structures.

With their data structures, neither Warnier-Orr nor Jackson intended to define the database but, rather, the structure of programs, and most of the bibliography we had available referred to batch programs. But their work was a most valuable resource in representing our visions of data.

Thirdly, a procedure is necessary to change from a set of data visions to the corresponding model.

We are now in the world of math, and the question is: ***For a given set of data visions, is there a minimal relational model to satisfy them?***

Finally, we now have all questions relating to this matter, or better yet: *we have rigorously formulated the problem*.

This state allows for a great number of tools that can be of help in solving it.

In this specific case, in addition to the usual computer tools, we have also used techniques and tools taken from mathematics, logic and artificial intelligence.

The work done led us to solving the problem. An important byproduct was to enter a new and promising world: that of quick prototyping.

We have frequently resorted to prototyping in our research work, to then make its use viable for clients, based on GeneXus.

At the beginning of this project, we didn't think of generating programs, but the regrettably, well-known "stable databases" continued to fail and led us to think that, sooner or later, we would have to face and deal this issue. Otherwise, our clients would always be under the pressure of high maintenance costs.

We were no experts in the automatic generation of programs, but the topic was not new at all:

From almost the start, computer science has tackled the problem of the automatic generation of programs.

For the longest time, and for the most part, program generators were quite primitive and most of all oriented at generating simple reports based on flat files.

After 1985 there were significant advances in this field and generators started generating both batch programs and transactional programs, and interacting with databases.

Those generators were based on templates, acting as "skeletons" which, starting with a fill-in-the-blanks approach, then became increasingly sophisticated, to end up solving a reasonable part of the needs for one installation to be set up.

It was clear to us that the automatic generation of programs would become, sooner or later, a fundamental matter to deal with.

The Million-Dollar Question: Are stable databases possible?

“Stable databases” are a recurring issue in computer science. The idea was presented this way:

> If we come up with the “right database” for a specific firm, then that database will remain stable in the future. Consequently, we will save time by writing programs that will use that base.

If that were not possible, then we wouldn’t have been able to come up with the “right data-base.” A lot has been written about this subject.

But the basic thesis is in fact false! The only way for an organization to have a stable model is when the firm has become stagnant or has died.

Therefore, it’s best not to waste efforts on searching for such stable models, but rather to work on possible, real, unstable models.

If we analyze it, we can break the question down into the following questions:

- How can we reorganize the database when it undergoes structural changes?
- How can we modify programs so that they can function properly with the new database?

The first question takes us to an in-depth study of how to convert the regular contents of a database with the old structure into new contents with a new structure.

The initial theoretical problem is: ***Can such conversion take place without data loss?***

If the answer to this is affirmative, then further questions will come up, like: ***What must we do to carry out the conversion and prevent data from being lost? And could we generate programs automatically for performing such conversion?***

The answer to the second question is obvious: If we are capable of generating the programs then why not generate all programs again?!

Resorting to brute force will almost always take us to an initial solution. But when we are faced with thousands of programs, that doesn’t seem to be the best choice, even considering the great, permanent increase in hardware power.

Clearly, this answer isn’t good enough and, in turn, leads us to another question: ***With changes to the database, could we determine the programs that will be affected by it in order to re-generate them?***

If we manage to achieve this it will be great, but the scenario brings about yet another question: ***Are there programs generated for the old structure that function correctly with the new one, and that are capable of being replaced by other more efficient programs?***

On a Sunday in August 1986, in New York, when we were still far from coming up with a company and a product, and focused on putting together a number of scientific and technological discoveries under a license to be offered to others, we discovered something aesthetically fantastic: **“extended tables”**

We won’t go into this topic in detail here, with which the GeneXus Community is quite familiar, because we’re focusing on questions, not answers. But briefly, we’ll highlight the following few important points:

- What is an “extended table”? For each record in a specific table there is a virtual record made up of the concatenation of the original record and all records of other tables in the database that are directly or indirectly defined by it.

- All those records will constitute what we call an “extended table” of the original table, and the latter is called the “base table” associated with the extended table.

How are extended tables significant?

Descriptions expressed in terms of extended tables remain in force through structural changes in the database.

Consequently, even when certain programs are actually not correct or optimal due to changes in their database, the descriptions of the GeneXus user visions remain valid. And this makes it possible to automatically propagate the changes by identifying programs that are not valid to generate them again based on the original descriptions of their visions.

Taking The Road Less Traveled

We never had a single doubt about this discovery being aesthetically elegant and, as time has passed, we’ve been able to prove in a number of ways that it is far more than that.

Our intent was originally to sublicense the technology to the big players in the field. But reality proved this idea to be quite naive: ***How much credibility could such advanced, new ideas have for huge tech companies in developed countries, when they originate in a territory that hasn’t traditionally had substantial production in the technology?***

The dilemma was between making our discovery public and putting together a company and a product to take a shot at marketing it, starting in Uruguay, and then move on to other destinations worldwide.

We went for the second option. We thought that, over time, if our product was actually useful to clients and our customer base grew consistently — even if at a slow pace — and if we managed to duly support new technologies, we would succeed.

So we were profoundly committed to our new purpose and, by late 1988, we founded GeneXus, we named our first product “GeneXus,” and we set out to launch its first version in the second half of 1989.

What were the platforms that GeneXus generated applications on in the beginning?

The realistic attitude was to initially choose a single platform and then specialize on it until we could come up with a minimal volume of business that would empower us to undertake new platforms. Our choice was IBM AS/400.

In which platform did GeneXus function at the start?

What we needed was an efficient platform that wouldn't require a huge financial investment to maintain its development.

We also sought the greatest independence possible from the platform for which we would be generating applications, so that we could then make the generation for other platforms something viable.

Our decision was to go with PCs with DOS operating system.

What are the problems that GeneXus needed to solve on the spot, and which ones could be postponed for later?

It was clear to us that, in the beginning, we would grapple with a lack of theory and practical inconveniences that would prevent us from generating full applications.

One of our objectives was to generate as much as we could, automatically.

Our second objective in mind was to automatically maintain everything we generated.

And this was the reason for totally discarding the generation of parts of programs for developers — because what developers would write manually would not be automatically maintained.

We did not know how to describe procedural programs like batch processes or casuistical routines that could not be derived from transactions.

There was the chance of introducing a 4th generation language to solve all those problems, but that would mean giving up the automatic maintenance of these programs, so we decided not to go for that.

We opted for implementing that which could be automatically generated and maintained, and without restrictions, like transactions, queries and simple reports.

We calculated GeneXus to automatically generate and maintain 70% of the programs in an installation, while doing the automatic design, generation and maintenance of the database. The rest would require long hours of additional research.

Could we automatically generate and maintain 100% of applications?

That was our next goal.

Generating and maintaining 70% of programs automatically seemed to be a high degree of success. There were only a few generators capable of coming up with similar solutions, but none of them could provide automatic maintenance. That is where we excelled.

One thing we had no doubts about was the fact that automatic maintenance was a very important and unique feature of GeneXus. But we believed that clients chose GeneXus because of the significant increase in development productivity, while automatic maintenance was something seen as an added, unexpected new feature towards which most of GeneXus outsiders were skeptical, to a certain degree.

But we were soon surprised when several clients shared their thoughts to us:

> "To automatically generate 70% of programs will be of great help and that is something we appreciate a lot."
>
> "Having to manually write 30% of programs is a restriction we accept in the belief that you will surely overcome this in the future."
>
> "But what is not acceptable to us is having to do the manual maintenance of programs not generated by GeneXus."

How could we satisfy our clients? How could we be sure of generating full applications?

The answer: By means of a procedural language.

Our question was: ***Can we build a procedural language with descriptions (source programs in that language) that will not become invalid in the event of structural modifications to the database?***

The answer was YES, and this enabled us to release the first full GeneXus.

Then questions started to pour in, endlessly…

- Can we generate applications for other platforms and architectures?
- Can we identify patterns in our descriptions?
- And based on that, can we automatically generate GeneXus objects to fit those descriptions?
- Can we extend GeneXus beyond what GeneXus itself does? Can we grow the GeneXus Community to make this possible?

… and many, many more questions….

Asking new questions is a constant occupation for us in which there are an increasing number of people taking part, like the research and development team, all of GeneXus, software houses, clients, and the entire GeneXus Community.

Through the years, we have encountered endless questions, most of which generated a lot of answers, and quite a few that helped to shape our product.

We can summarize all of the questions into one main principle:

We can "describe" instead of "program"

and one strong belief:

We must never lose our freedom of thought!

Acknowledgements

We'd like to particularly thank a number of individuals and companies: our team, our clients, and the whole GeneXus Community.

And also, we would like to humbly express our appreciation to those who, despite not being clients or participants in our Community, were somewhat skeptical about GeneXus, for continuously presenting new questions and challenges that are our inspiration for further learning.

We have a wonderful world, and anything capable of nourishing our knowledge of it must be seen as a privilege and a unique opportunity!

About the Authors

Nicolás Jodal, Chief Executive Officer at GeneXus

Nicolás Jodal is a Systems Engineer graduate of the School of Engineering of the Universidad de la República (UdelaR). He has been a Professor at the Universidad Católica del Uruguay.

He delivered numerous courses within his specialty field of study in Brazil from 1984 to 1989.

From 1984 to 1986 he provided consulting services in database and application development projects to several of the largest Brazilian and Uruguayan companies.

He is the co-author of the GeneXus project. Together with Mr. Breogán Gonda, Mr. Jodal was distinguished by the National Academy of Engineering (Uruguay) with the National Engineering Award 1995, because of the GeneXus project.

His areas of research include: databases, artificial intelligence, automatic application development methodology and interaction between the business and information technology worlds.

Mr. Jodal is Founding Partner and Director of GeneXus and GeneXus Consulting in Uruguay and GeneXus, Inc. in USA, GeneXus de México, GeneXus do Brasil and GeneXus JapanInc. in their respective countries.

Breogán Gonda, Chairman of the Board at GeneXus

Breogán Gonda is a Systems Engineer graduate of the School of Engineering of the Universidad de la República (UdelaR).

He has been a Professor in the School of Engineering of the Universidad de la República (UdelaR), in the Pontifícia Universidade Católica de Porto Alegre (Brazil) and in the Universidad Católica del Uruguay.

Within his line of studies, Mr. Gonda has delivered courses and seminars as Visiting Professor for post-graduates at universities of several Latin American countries.

From 1976 to 1989 he has provided consulting services in Database and application development projects to several of the largest Brazilian and Uruguayan companies. Mr. Gonda has delivered numerous courses within his specialty field of study in Brazil from 1976 to 1989.

He is the co-author of the GeneXus project. Together with Nicolás Jodal, his partner in the GeneXus project, Mr. Gonda was honored by the National Academy of Engineering (Uruguay) with the National Engineering Award 1995 because of their work in this project. He has been acknowledged by the Uruguayan Association of Engineers as “Outstanding Engineer of the year 1996”. In July 1999,

Mr. Gonda was appointed a Member of the National Engineering Academy (Uruguay).

His areas of research include: databases, artificial intelligence, automatic application development methodology and interaction between the business and information technology worlds.

He is Founding Partner and Director of GeneXus and GeneXus Consulting in Uruguay and GeneXus, Inc. in USA, GeneXus de México, GeneXus do Brasil and GeneXus Japan Inc. in their respective countries.