

Necessity they say is the mother of invention - it certainly was in this case. The origin of the conceptual ideas behind 'GEM Networking', go back to work that GEM's System Architect, Brian McGuigan, did whilst Operations Research Executive for Tasman Forestry in New Zealand.

Whilst there, Brian successfully developed a large-scale linear programming model that told the company: when to thin and clear fell its forests; which processing sites to send the logs to; which processing plants should process them; where the processing residues should be sent; what products would be produced from all the sawmills, particle board plants, pulp & paper plants and so on; and which markets to sell them on, in order to maximise the value of total group cash flows for the next 60 years – quite an undertaking and far more complex than GEM!

This was one of the most complex optimisation models ever developed, not just in New Zealand, but in the World. So complex, that when the project began the company realised it did not know what it needed to know, to solve the problem. This meant the conventional approach to computer systems design could not be used.

Usually you specify what you want a system to do. Define how you are going to solve the problem and then the information required to solve it. They certainly did not know either how they were going to solve the problem – let alone what information was needed to solve it!

Brian therefore realised they needed a highly flexible approach to software development. That is quite a tall ask, because when you design software you design it to do a particular job. Software is usually pretty good at doing what it's designed to do, but no good at doing things that it's not designed to do – obviously! So usually the words 'flexibility' and 'software' are at opposite ends of the spectrum.

They did however realise the things or 'Entities' they needed to model : Crops, Processing Plants, Locations, Transport Routes, Resource Types (Logs, Residues, Finished Products and so on), and Markets. They therefore set about building little independent models of each of the 'Things' they knew they would need to model. These resembled little independent spreadsheets - only they did not call them that, as spreadsheets had not yet been invented! They then developed a way of readily adding new kinds of information to any of these 'Entities' and a generalised way of extracting the information from them and presenting it to the model for solution. They also had a model generator that enabled them to specify the mathematics of the problem in a special purpose high-level language. The model-generator then generated and compiled a program for them that solved the problem.

Using this approach they cut down the time it took to identify an unforeseen problem with the model, re-do the necessary mathematics, decide what additional information they needed to resolve the problem, add it into the database structure and regenerate and test the model, from two months using conventional hand-coding techniques, down to one and a half days – on average. This was invariably faster than the users could come up with the data required to drive the new feature. So it did not matter, as far as the users were concerned, whether the model could handle every aspect of the problem or not, as it could always be modified so fast it never held them up. (One user likened it to modelling in plasticine, rather than granite.)

Using these techniques Tasman Forestry's team of 1 ½ people actually succeeded in solving the problem, whereas they heard of a team of 35 people in the US who gave up on the problem after 3 years of development work because it was 'too complex'.

Brian realised that one of the reasons for their success was because their ignorance had forced them to design their database from the 'bottom-up' - based on things that actually existed in the real-world. This meant they always had somewhere that they could plug new information into. So they hardly ever needed to move information to different locations in the database. This, in turn, meant that they hardly ever needed to modify any code that had already been developed that relied on finding data in its old location. All they ever need to do was, add new information and develop new code! (Coders Nirvana😊)

Thus it was that Brian had these concepts in the back of his mind, when, having left Tasman Forestry, he was looking at all available CRM's to select one to market his own computer services company. He was of course disappointed to find that none of the available systems had an adequate underlying data model. This was because, living in a small town, he often came across the same Organisations and People in different roles, so wanted a database structure that reflected this reality. He also wanted a system that you only needed to make a single change to, and everything that was affected, was done automatically. He did not want to have to remember what else needed to be changed – realising that this was an impossible task that was bound to lead to errors.

So he began to think "I can do better than THAT!" Initially he started messing around in the evenings with the concepts involved. The more he did, the more excited he became about the possibilities and decided to turn it into a product that his company could sell, rather than just a system for their own use. The rest, as they say is history.

Being based on the concept of generalised 'Entities', Brian decided to call the system ' GEM Networking' – GEM standing for '**G**eneralised **E**ntity **M**odel' and 'Networking' because that is what the system is superbly good at.

Since the system also uses generalised 'Relationships', Brian briefly considered calling it a 'Generalised Entity Relationship Model'. He quickly realised that this was not a good acronym for marketing purposes - GEM seemed a much brighter idea! Nevertheless Brian expects GEM to 'catch on' just as fast anyway. 😊