OOP 2001
Architecting Large Business Systems

Case Study and Excercises: The Nerdy Geek Company

Alan O'Callaghan, De Montfort University
Jens Coldewey, Coldewey Consulting
Wolfgang Keller, Generali Vienna Group

Munich January 22nd 2001

Alan O'Callaghan
De Montfort University
The Gateway
Leicester, LE1 9BH
United Kingdom
aoc@dmu.ac.uk

Jens Coldewey
Coldewey Consulting
Curd-Jürgens-Str. 4
D-81739 München
Germany
jens_coldewey@acm.org
http://www.coldewey.com

Wolfgang Keller
Generali AG
Kratochwjlestr. 4
A-1220 Wien
Austria
wolfgang_keller@acm.org
http://www.objectarchitects.de

1 The Case of Nerdy Geek Ltd.

1.1 The Company

The NerdyGeek Company was set up in 1985 to bring arts and crafts from Bali for sale in what was initially a single retail outlet. NerdyGeek has since grown to five retail outlets, all of which are very successful, and has a perspective of further expansion.

Originally, NerdyGeek simply bought up available items from the existing stock of craft suppliers in Bali and placed them on the shelves of the NerdyGeek shops. Shoppers would walk in off the street, browse the displays and purchase items that they wanted with cash. As this casual trade strengthened so that NerdyGeek began to see regular and frequent customers two things happened: NerdyGeek began to take credit card payments and also established a Purchasing Department so that popular items could be ordered in advance from preferred suppliers. A central, custom-built warehouse was established to store stock so that when the shop ran low on a particular range it could be quickly replaced.

It wasn’t too long before NerdyGeek’s regular customers would start to phone in to the shop and order an item in advance. In order to support this service NerdyGeek developed a system by which such customers could arrange to pick the item up from the shop on a certain date .

NerdyGeek currently has six departments besides a core corporate administration function which deals with corporate policy, legal and tax matters, personnel etc. These other departments are:

- Purchasing – orders stock, tracks storage and estimates warehouse delivery dates
- Retailing – takes and tracks customer orders, controls the retail outlet operations etc.,
- Warehousing – storing, picking etc.,
- Shipping – arranges and maintains shipping schedules
- Receiving – checking of goods inwards against purchasing orders etc.,
- Accounting – customer payments, stock values, payroll etc.,

1.2 NerdyGeek’s Requirements

NerdyGeek’s operations have outgrown the computing system installed several years ago to support them. Designed using structured methods, the system is essentially a (relational) database together with a number of programs that access it. The current system lacks certain functionality that has been important to NerdyGeek's operations for a while now. In particular customers who place orders are unable to get hard information about stock availability (if the item is not already in the shop) or price while credit account customers are unable to get prompt replies as to payment status etc. This is partly because the system is not a distributed one – it sits at NerdyGeek's headquarters and has no direct interface with the retail outlets or the warehouse for example – but also because it was designed to support a rather different kind of enterprise with rather simpler business processes than those currently operated by NerdyGeek. NerdyGeek therefore wants to replace the current system (without losing current data, of course). They also want to future-proof the new system as far as possible. The experience of trying to maintain the old system in the face of changing requirements has been a painful one. They have come to you, renown experts in object-oriented development and component-based development, because they have heard that the utilisation of these technologies can ease the maintenance nightmare.

1.3 Possible Extensions of the System (needed for 3rd exercise)

The following scenarios are selected results of applying the "Buffer the System with Scenarios" pattern. They are contained in a strategy study the NerdyGeek management handed out to you. You need this information for parts of the third exercise.

1.3.1 Scenario: Remote Delivery Order

In the future it might be advantageous and profitable for NerdyGeek to extend the ordering service it currently supports in its retail outlets. Indeed in the face of emerging competition it might prove a necessary step to improve customer services.

Customers could place orders for home delivery direct from the warehouse (currently they can only place advance orders for collection from the shop). The order would still be placed in the shop (in person or by telephone) but would be despatched directly from the warehouse on an arranged delivery date. Shipping charges and taxes would probably apply to the price.

Possible implications: it might be appropriate to consider a 5% discount on price to those who choose to collect advance orders from the shop in order to maintain the relevance of the retail outlets and strengthen the customer-retailer relationship

1.3.2 Scenario: On-line Ordering

It seems unlikely that NerdyGeek will be able to avoid the internet for much longer. In particular the possibility of customer's making on-line orders should be considered. Internet users should be able to browse an on-line catalogue, check stock availability and prices and, by opening a credit account based on credit card debiting, place an order online for home delivery.

Possible implications: there is a major impact on customer-facing business processes because, for the first time, the customer's point of contact will not be with the retail shop.

1.3.3 Scenario: More Floor Space

If current expansion trends continue it will be necessary to acquire more storage capacity. Currently NerdyGeek has one, central warehouse with a standard storage allocation policy. It would be foolhardy in the extreme to build another warehouse at the moment because there is no guarantee that the expansion trend is the kind of permanent one that will justify heavy capital expenditure. The alternative is to lease out floorspace from warehouses owned by storage companies.

Possible implications: the storage allocation policy may not be enforceable in leased warehousing space because of rules applied by the warehouse owner. Rather the policy that will be applied is that of the warehouse owner, or one constrained by that policy.

1.3.4 Scenario: Precious to Keep

There is a growing trend of demand by customers for larger, more expensive products from Bali (e.g., hand-carved furniture). While not an issue at the moment, it is possible to foresee the need to store some special items in a non-standard way.

Possible implications: For some items there may need to be a specific storage policy, unique to them, which is currently not catered for in the warehouse's standard storage allocation policy.

2 Exercises – General Remarks

- The time frames for the exercises are challenging! This is part of the process. Stay focused on the result and don’t waste time discussing details, methodology or technology.

- Your tutor is there to help you with any question you might have. Feel free to call him. Additionally your tutor will do “management by walking around”.

- You will notice that you come from different working cultures. Be gracious to each other and be prepared to clarify misunderstandings. This is the perfect setting to try something new!

- To make the process more effective we urgently suggest to **assign a moderator** whose sole responsibility is to facilitate the discussion and solution process. She or he should also keep an eye on the time budget. Assign a different person to document results during the process either on paper or with your favourite tool (text-processor, mind-mapper, CASE tool, drawing tool, Solitaire,...)

- There will be a shoot-out process after the first two exercises to find a common solution for your tutor’s group on which to build up in the next exercise. Details will be explained. Be prepared to send **two delegates** of your group to defend your results in the shoot-out. Also, be prepared to loose the shoot-out and use a different group’s result. Please observe the shoot-out closely to understand all presented suggestions. Of course, we also have fallback solutions just in case you don’t like any of the solutions during the shoot-out.

3 Exercise: Mile Wide Inch Deep

You are the architecture team of the NerdyGeek project. Your job is to develop a first-cut, top-level domain architecture. The client wanted to have it yesterday but granted you another 30 minutes to do your job. If you don’t have a result after these 30 minutes your competitor will get the contract and you will be fired by your boss (at best).

Identify actors and main business objects and their relations

Try to find a suitable top-level package structure. How do the packages depend on each other? Which dependencies are critical?

Do not waste time on technical architecture. Assume you have a single server, silver-bullet internet platform in an ideal world.

4 Exercise: Organisation Follows Architecture

You managed to present the architecture after 29 minutes and thus won the project in the first shoot-out process. Now you have to organise the project. The team will have 16 members and all are generalists, being able to do any job[1]. You have 25 minutes to set up the team or your manager will assign the people to a different and very important project and you have to do the job on your own.

You are using a fast lightweight process, every member of the team will pick up chunks of the design work later.

Which roles do you need?

Is there a reasonable sub-team structure?

Turn the roles and the team structure into an anonymous org-chart with 16 people to be defended in the shoot-out process. Be aware: You will work in this structure in the third exercise! (Hint: Reflect the optimal team size to start with. What are options to get there?)

[1] Dreams are my reality...

5 Exercise on Design: Putting Ideas Into Reality

Now your project starts. Delivery is next week, including coding and testing, so you don’t have too much time to do the design. In this exercise we concentrate on design issues that are relevant to the architecture: Decoupling packages and ensuring extensibility. But remember: Do the most simple thing that might possibly work!

We suggest to use the team structure you developed in the second exercise.

Probably your package architecture still contains relations that turn out to be cyclic. Find a design that breaks these cycles and establishes the package dependencies defined in the architecture.

Consider the extension scenarios from chapter 1.3. Design the appropriate “extension hooks” that grant the needed flexibility without adding too much complexity.