The software logistics tail

Have you noticed that when you start coding you can move fast, laying down lots of code in a short time, but the more code you lay down the slower you're rate of progress becomes? Likewise when we think of a new feature we sometimes get the code written quite fast, but other times a relatively small feature takes an inordinate amount of time. Sometimes we get bogged down the practicalities of life checking in checking out, building, fixing link problems, sorting our machine, dealing with ripple effects and so on.

Software development has a logistics tail, just like an army advancing we need support, to move forward requires logistics support. When an army advances the assault troops at the front of column need to be constantly supplied with fresh ammunition, food, medical facilities and countless other items. History provides many examples of armies that have advanced beyond their logistics support - or failed to push home their advantage for fear of ruining ahead of their logistics tail.

Paratroopers can capture objectives but they can't hold them alone. Assault troops may look light weight, fast and agile but they won't win the battle alone, they need heavy support and logistics.

In software development the logistics tail contains source code control systems, build systems, testers, system administration, database administration, release mechanisms, customer support, fault tracking systems and so on. Nor is it confined to activities, the logistics tail exists inside our source code too.

We have to consolidate our gains if we are not to lose them. A new feature may be demonstrated quite quickly, but we must ensure it is placed within a well-defined program structure, we need to give it a place, safely embedded in a library or DLL, not hacked onto some odd ball Windows message. It is not enough to capture a new feature, we must secure is before we can move onto the next objective.

If we have some tricky code we treat it like a minefield, we don't hide this irritating detail in the hope that nobody will encounter it. Instead we **WRITE IT BIG**, with a great big sign saying "Danger!" – not just a comment, but an assert, or better, a static assert, or maybe we ensure code can only be called on the safe-way.

Having a well-defined and reliable logistics tail means we can move forward with confidence. Imagine working without a source code control system, imagine working "manually" – flipping the "read only bit", slowly making changes to different files, manually finding out who has the latest copy of any particular piece of code, distributing our results – the problem is exponential in size. So while setting of a CVS server may seem like distraction from the objective it is an essential part of securing the territory.

Of course this military analogy may seem somewhat dramatic, nobody ever shot a software developer for failure to deliver on time, but dramatic ideas that allow us to visualise, and help produce a clear, communicable idea.

May be, a few of you reading this, will be thinking: "I want to be the SAS of software development, get in, do the job and get out fast" - nice idea, and perhaps something to aim for. But don't forget that meticulous planning that goes into such operations and even the SAS couldn't operate without a logistics tail of support staff, helicopter pilots, ground crew, training and special equipment. Perhaps ironically, the logistics tail becomes more important if you want to be truly responsive and agile.

(Of course it helps if you know how to implement polymorphism in 15 different ways of your bare hands.)