Chapter 7 – Conclusions and Recommendations

W. Morven Gentleman

Morven.Gentleman@dal.ca

"Sunny-day" systems, that fail to take into account predictable things that might cause the system to fail, are inadequate and all too common. Ignoring consequences of fallible construction is a perfect example. Understanding of software dependability needs to be ingrained in all stakeholders, not just system designers:

- Robustness needs to be built-in, and cannot be an add-on.
- Implementation faults (bugs) are only one concern, maybe not even the most critical.

Software dependability cannot be achieved by middleware alone. Robust components are nice but also not enough.

Systems are made up of hardware, software, and people. Processes (i.e. workflow processes, not just the operating system software process abstraction) are critical, as are tools to support the processes. In the event of failure, systems need to be able to continue to operate, albeit in degraded mode, and ultimately to return to full operations status: graceful degradation epitomizes robustness. Procurement needs to take a much more pro-active stance on requiring robustness and recovery. Operations needs to plan for, and practice, recovery from failures. The inevitability of human failure needs to be recognized and planned for, not just in the form of end-user invalid input and misinterpretation of output, but also in incorrect actions of operational staff as well as support such as system installers and maintainers. Particularly insidious are data corruption faults that do not immediately give indication of failure, but may lurk undetected for days, weeks, even years. Ultra-large scale systems have highlighted problems that traditional approaches don't solve. Nevertheless, some of these problems have long existed even in smaller systems, such as systems-of-systems.

Recovery-Oriented Computing is different from Disaster Recovery. The latter is typically taken into account by operational units as addressing business continuity and business resumption in the face of cataclysmic, but exceedingly unlikely events such as national power or communications outages, or aircraft crashing into computer centers or in the military context severe battlefield reverses. These vulnerabilities are indeed wise to consider, but are unlikely to warrant investment in automated response. Recovery-Oriented Computing, on the other hand, investigates automated aid to assist operational staff in restoring service after unavoidable events for which the risk is enough that the investment is prudent.

Although research on software fault tolerance has been done for almost 40 years, more needs to be done. This is definitely not polishing a shiny jewel, but rather addressing problems not yet resolved, problems of direct relevance to today's net-centric systems. Funding must be found for this research.

The original plans for this workshop proved overambitious. Significant results were achieved, so this is not a criticism in and of itself, but it stresses the need for continuing efforts.