Report on the EuroPLoP 2003 Lego Mindstorms XP Development (with patterns) Challenge

by
Alan O'Callaghan and Maria Kavanagh
(aoc@dmu.ac.uk; kavanagh@dmu.ac.uk)
Faculty of Computing Sciences and Engineering
De Montfort University
The Gateway
LEICESTER LE1 9BH

June 2003 and the eighth European Pattern Languages of Programming conference may yet prove to be the most historic. On the Friday there emerged the first activity of an extremely ambitious experiment which, it is hoped, will run over twelve months and more. If the experiment is successful it may reshape the thinking of the Patterns Movement. The aim of the experiment is to validate the feasibility of a truly generative pattern language (in the Alexanderian sense) for computing intensive systems. The difficulty of such an experiment is the explosion of design variables that emerge in any non-trivial constructed system, and therefore the size of any pattern language that could accompany it. Christopher Alexander, let it be remembered, considered patterns to be rules for making structures that were in some sense 'living'. A pattern is both the thing and the process for making that thing. The trick is to organize all the appropriate patterns to make all the 'things' in your system in such a way that the 'whole' they construct and the different elements that make up that whole exist in harmonious relationship with each other. This implies that the pattern language constitutes more or less the whole of the overarching process by which a system is created. Or to be more precise, it is by applying a *sequence* of patterns, each one being applied to the system and thus changing the context ready for the next to be applied, that the system is constructed.

How to explore this? This was the challenge that the organizers of this Focus Group (which was how space was found for it in the conference) faced. What kind of non-trivial system can be found that can be scoped sufficiently for a laboratory experiment? What patterns can be found for this domain? How can their efficacy be tested?

The answer to the first question came in a ready made workshop developed originally by some folks at Connextra called the Lego Mindstorms™ Challenge (a.k.a., *The Yellow Brick Road Challenge* in due homage to the Wizard of Oz). They had used it in fifteen or so conference events around Europe to demonstrate a subset of eXtreme Programming practices. Teams of five people were given two sets each of Lego Mindstorms™ robots and divided into hardware (2 persons), software (2 persons) and 'Tracker' (one person who policed the XP rules) subdivisions. The challenge was to build a robot that could run three times around a 'track' described by a thick, black oval line on a white mat, in the fastest time. Stories were used to develop requirements and to plan iterations. Roles had to be changed after each iteration so that everyone played at least the software and the hardware roles.

Friday 2.30pm saw the workshop start in the Art Studio. Twenty bug-eyed volunteers itched to get their hands on the robot parts. The plan was that after a short fifteen minute introduction the teams would spend another 15 minutes choosing a metaphor. The next 30 minutes were to be spent in the Planning Game and planning the first iteration which followed immediately after. A stand up meeting was supposed to end each development. Subsequent Planning Game segments were to last only ten minutes with the stand up meeting/development periods remaining a half hour long. This allowed three iterations on the Friday. A further and final iteration was allowed therefore on Saturday morning before a 15 minute 'race' session and an hour or so for reflection and brainstorming.

Of course, in the long view, there was no chance to create or test a candidate pattern language in such a short time. Thumbnails of some thirty hardware construction patterns were handed out as a resource – participants also had available to them the Lego *Constructopaedia* and the online help of the proprietary programming environment – but from the beginning the aim of this workshop was to identify candidate patterns rather than validate them. This could be done at later workshop, perhaps at EuroPLoP 2004. To this end the workshop was fully documented in digital stills and video for 'post-match' analysis. Also filmed were end of the first iteration interviews with the trackers of the teams.

Needless to say, the timetable suffered the expected fate of all blueprints and masterplans in the face of deadline pressures – they crumbled (Long live piecemeal growth!). In the end, the group that maintained as many 'agile' practices as it could won the race, but no group fully kept to the rules. Indeed in the recorded interviews at the end of the first day, one 'Tracker' asked what progress his group had made replied 'None'. His group had effectively become amoebic and had divided to form two separate cells working on completely different robots while one member, having occupied the programming seat with the greatest control of the keyboard was never to give the seat up again. This group proved to be, sometimes amusingly, completely dysfunctional – driving its one woman member to complete distraction within the first half hour, for example, because her ideas were being ignored by the boys. Nevertheless, despite the tracker's assessment at the end of the first day one of the two robots seemed to be close to being finished. Its turning circle was too wide, but assuming that the software could be modified, that was something that could be fixed by morning – or so we all thought. Unfortunately, some one of the team wiped out the working program in the first few minutes of the next morning and the team was back to square one.

The human dynamics were amongst the most interesting features of the workshop. Mini tantrums were seen in one, generally successful group, with one software person 'taking a walk' and seeking the fresh air of an open window to cool down his frustration at not being able to test out one idea before someone else in the team started barking instructions about a new one. Nevertheless, even by the end of the first day each team had at least one (imperfectly) working robot.

What was very interesting from the patterns point of view were the common problems that were emerging. It became clear quite early on that the infra-red towers that communicated with the programmable Lego RCX 'bricks' had no means of identifying them uniquely. Consequently one team's beamed instructions often interfered with another team's robot's programming. Everyone attempted versions of the classic line-following robot using the supplied light sensor. They tended to vary in wheel design. The issue this raised was that, depending on the design, a small change in orientation of the robot (typically steering involved stopping or slowing down one wheel) caused a big change in the position of the sensor, often taking it well outside the area where it might 'expect' the black line to be. This was often followed by the robot spinning madly around itself in a desparate search for the lost line, or alternatively, a gentle meander in a heading that every millisecond took the robot further away from the line. Calibration was another issue as what the sensor decided was 'white' and 'black' varied as the robots were moved from their test tracks to the final race track.The visual programming environment was interesting, but apparently as confusing to sophisticated, educated programmers as transparent it is to nine-year olds who've never seen an IDE before! Either that, or as someone noted in a subsequent workshop in the series, everyone had reverted to junior age and become completely refascinated by the coloured Lego bricks. No group really looked for software solutions to the problems they encountered, instead they focused on hardware choices.

Five robots eventually completed the race. But not before last minute problems. Even as the cry 'Stop now!' rang out for the last time, the pigeons belonging to the dysfunctional group came home to roost when one member of one half-team stepped away in frustration from the programming console with the required synchronicity to completely crush the other half team's finished robot just as it successfully completed its last test lap. At almost the same moment another team decided robustness testing meant dropping their finished robot from five feet or so onto the hard floor. Kindly focus group organizers gave extra time to allow for last minute maintenance.

George Platt who, together with his Art Studio team was responsible for the photography, videoed the final race. Excerpts of this were shown in the feedback session to the rest of the conference on Saturday evening. Four robots were shown successfully completing the three circuits of the oval track at various speeds, and with different degrees of control. As it turned out the winning team were the ones who, until the last minute, had feared they would have no robot at all to race. They then found 'near solutions' in the official Lego documentation and had come up with the best design. Another group produced two, quite quick, but very different robots. An attempt to merge their apparently superior control software with the first team's speedier robot proved to be a failure however reminding us all of the configurational issues of system design (and in its own way demonstrating the non-trivial nature of the software).

The final piece of video tape showed the repaired robot of the dysfunctional team. Imagine a camera shot looking down on a black oval track. But there is no movement, no sound. No sign at all of a robot. Has the tape stalled? Five, ten seconds…Are we looking at a still? Fifteen, Twenty seconds… And then, all of a sudden, from the bottom right-

hand corner comes the robot. At speed. Going like the clappers in fact. No doubt it is the fastest robot we've seen. But its going in a straight line! In a flash its gone from bottom right to top left having cut the screened area neatly in half. This is unfortunate because the effect of its 'line retrieval algorithm' is missed and the nature of the programming is given no clue. And what a pity, because a few seconds after disappearing from view it is back again! Once more at top speed the robot traces another diagonal. Top left back to bottom right this time as it traverses, in the opposite direction, its original path. The achievement of the dysfunctional group was rapturously applauded by the entire conference. Here was a true icon of twenty-first century software engineering: a brilliant solution – unfortunately to the wrong problem!

The overwhelming feedback from the participants was that they enjoyed the entire workshop hugely. Even those who had negative experiences felt they had learned a great deal. The pattern thumbnails and even the XP rules were to a large extent ignored in the workshop, but the problems in hardware and software, as well as issues of human interaction, that future patterns might address were sharply revealed. Subsequent workshops have been held at vikingPLoP and at De Montfort University's Software Architecture laboratory. By the time you read this, new versions will have appeared at the Association of C and C++ Users conference at Oxford, UK and the rOOts conference in Bergen, Norway. A candidate pattern language called the Bots and Pieces pattern language has emerged and it is hoped it will be ready to test at EuroPLoP 2004. Watch this space (if you watch long enough that robot might come back!).

AOC and MJK 03/04