A software methodology for applied research: eXtreme Researching

Olivier Chirouze¹, David Cleary² and George G. Mitchell³

¹Sélatec, Lyon, France
²Ericsson Applied Research Laboratories, Athlone, Ireland
³Department of Computer Science, National University of Ireland, Maynooth, Ireland

ABSTRACT

Applied research is, by necessity, a distributed, collaborative process. To be useful, research methodologies must not only be applicable in such an environment, but must also be adaptive to the needs of human resources and specific research area requirements. This paper introduces eXtreme Researching (XR), an adaptation of eXtreme Programming (XP) by Ericsson, to support distributed telecommunications research and development. XR builds on XP and tailors it to meet the needs of applied industrial research. It adopts and extends the most useful elements of XP: collective ownership, planning game, continuous integration and metaphor and shows how they are applicable in multi-site, research projects. XPWeb is developed as a tool to facilitate XR in a distributed research environment. XPWeb and XR is used by Ericsson Applied Research and have been shown to significantly increase output and efficiencies in multi disciplinary research projects.

KEY WORDS: eXtreme Programming; agile processes; process engineering; telecommunications

1. INTRODUCTION

The software engineering community refers to the software process as the sum of all software engineering activities, encompassing the software life cycle from requirements, specification, implementation through to maintenance and future development [1]. A key characteristic of these activities is that they are subject to frequent changes and as a result must adapt or develop new processes [2].

A number of the methodologies used in software development today have their origins in the waterfall process [3] and tend to separate activities into distinct phases of design, coding, testing and integration. These activities have been found to occur repeatedly on each cycle or iteration. Iterative design processes [4] such as the Rational Unified Process (RUP) [5] have become widely used over the past decade.

As computing becomes more pervasive and ubiquitous, a major new research effort is required to develop adaptive information technologies that automatically evolve to meet the diverse needs of users, devices, and content in changing contexts. These developments will require new information and communication technologies; which we term Adaptive Information Technologies. These new ICTs will develop from novel advances and syntheses in both the hardware and software areas. Indeed, a key novelty of this applied research is that it will bring together diverse skill-sets and talents. This environment requires the convergence of many research themes. This new multi-discipline research calls for a new coordinated research approach.

The idea of rapid prototyping and researching is intrinsically linked, especially within a R&D department. One of the challenges for running a research project is to balance the structure of a software methodology with that of maintaining the creativeness of research. Selecting a software methodology that satisfies these requirements is not easy. With the development of the Agile Methodologies some of these problems have been overcome. Agile Methodologies focus on being people-oriented rather than process-oriented and are adaptive rather than predictive [6]. Thus they are adaptable and have been successfully adapted to meet a number of challenges not least an eXtreme Programming (XP) instance of RUP [7].

Problems associated with distributed research project methodologies continue to persist. Today a combination of professional researchers and academics carry out the significant majority of the applied research. In the area of mobile telecommunications Ericsson is always looking for that magic formula or process, so as to reproduce previously successful collaborations. The unpredictable nature of research software development led us to look at the family of agile methodologies. Among the many recent processes that embrace unpredictability rather than trying to prevent it, we realized XP was the one that has values that map to research the best; this is described in the next section.

In the rest of this paper we present an eXtreme Researching (XR) process, which we have designed. In Sections 3 and 4 we examine single and multi-site (distributed) XR and this leads us to Section 5 where we...
outline our core concepts and requirements for XR, a process specifically designed for applied research within a distributed framework. Section 6 introduces in detail the enablers necessary to operate a distributed XR project, highlighting the IT tools that are required. Finally, in Sections 7–9 we conclude the paper, presenting observations we have made while using XR, a discussion and the conclusions we have drawn from developing the XR process.

2. XP FROM A RESEARCHER'S PERSPECTIVE

Before discussing XP from a researcher’s perspective, it is important to define our concept of research activities. Research can be perceived as many things depending on your viewpoint, academic centric of industrial. In this paper we define applied research as an activity that gains intellectual leadership that ultimately leads to commercial reward for a company by combining the classical viewpoints. We measure this in terms of the combination of intellectual leadership (knowledge) together with the ability to demonstrate new ideas through proof of concept prototypes.

The use of XP is gaining increasing popularity, especially in the mainstream software industry, and it is not surprising that university and industrial research institutions are keen to explore the use of this process as well. One of Kent Beck’s recommendations is to ‘embrace change’ [8] and, in a sense, this idea is a binding philosophy in the research environment and enables the evolution of innovative technologies. It is for this reason that XP seems to be a logical choice for researchers to explore. XP is based on the sound values of communication, simplicity, feedback and courage [8,9]:

- XP as a methodology transparently enforces teamwork, placing a major emphasis on communication.
- eXtreme Programmers are encouraged to keep everything they do simple. Although it is harder to link this with how researchers write software, it can be argued that code is not the main resource of research projects. Thus, the code must be written fast and be easy to understand.
- XP practices aim to provide feedback that is constant. Feedback is probably the main value that is expected when building a proof of concept. The obsession of XP for feedback is definitively valuable to researchers.
- The last XP value is courage. This is an important requirement for any project, the courage to embrace change and make hard decisions. Courage to be realistic and honest when discussing ‘normal’ development projects with the customer is also required on research when exciting though not valuable solutions must be thrown away and issues must be re-thought from start.

Looking at the values XP stands for or embracing XP as a philosophical process [10], it is possible we are on the way to discovering the ideal research process.

3. RESEARCHING AND XP

Single site use of XP in a research environment has been examined in a number of recent case studies [11–13]. These studies have also shown that in a university research environment it was difficult to maintain adherence to the new XP methodology, for example the adoption of pair programming. Researchers have little or no experience of pair programming (the majority of programming even within a development team is generally performed by the individual alone) and as a result it has been reported that the coordination of pair programmers was difficult [12]. However, a recent single site scientific research case study involving two researchers [13] has reported that pair programming can be used and is very beneficial for researchers. Our experience has shown that in applied research the practices of XP can be adapted and while they can be challenging to implement and sometimes require tailoring, a significant benefit to tailored XP can be derived.

The following are the key attributes from XP that we think directly map to research and form the core practices and principles of XR.

- Metaphor: there is a strong analogy between the way XP describes the architecture of a system, and the way it is understood in research projects. This metaphor—an image that all the members can understand—is necessary to provide a high-level concept rather than just its implementation. Researchers usually do not have a clear definition of their “needs”, but rather an explicit description of a concept. We believe that maintaining a clear definition of this metaphor is beneficial.
- Development: the development cycles in XP are very short. An XP project is usually a series of iterations, each lasting 1–4 weeks. The way a system is built is also quite different to usual
methodologies. Using the practices of simple design and refactoring [14], code is kept simple but scalable. When dealing with complex concepts, researchers tend to keep their designs simple and to work in short cycles. In applied research projects, refactoring is used to go from an unclear idea to a proof of concept, the ideal scenario being that the research proof of concept results in a product offering being developed.

- Testing: XP takes testing to the extreme through the encouragement of test-driven development. A feature inclusion in a small release has to be justified by extensive unit testing that verifies that use case. Researchers are usually not so focused about tests. However, the concept of verifying each stage of a prototype is of vital importance when tackling a large research problem (project). Tests as they are considered in XP do not just test the code; they increase the design quality, describe the system and support other practices. The practice of robust coding does not map to researchers habits, but we have found that major benefits can be achieved by having the courage to adopt this discipline.

- Collective ownership: the practice of collective ownership is not normally found in a research environment. In XP, programmers can change any piece of code of the project. Researchers are more likely to be specialists in their domains, working on specific parts of the same project. This practice appears to be one of the most challenging to implement between researchers, we believe it is a key step to increasing communication.

- Discipline: the use of coding standards supports all the other practices, and XP reiterates the belief that following a sustainable pace is the only way to manage a project to the end. These quite obvious practices help when running a large research project.

These attributes from XP require effort but are valuable for research, whether the research is conducted in a single site or in a distributed fashion. This can be where it is distributed across a university campus, a software development company or even across a number of continents.

4. DISTRIBUTED RESEARCH WITH XP

A major difference between researchers and software developers is that researchers are domain specialists and are more likely to work on more than one single project at any one time [15]. They may have different activities and are routinely relocated abroad (collaborations, consultations, etc.). This relocation and multi-project development results in a situation where communication, a necessary part of XP, suffers. The establishment of single site development teams within a multi-national corporation (and including collaborations of multiple institutions [16]) is a challenge. Experience in a number of international companies shows that rather than trying to change the location of the project, the software process must adapt [17]. We have found that in our research setting, XP needs to be distributed. XP experienced by Wood and Kleb [13] faced similar difficulties to those we have faced. One significant difference is that Wood and Kleb’s projects were far smaller scale, involving only two people working locally.

The ability to cope with changes in development team structure and the issue of distributed location are two fundamental problems for XP. XP has attempted to address these in one of its latest incarnations, Distributed eXtreme Programming (DXP) [17]. Kircher et al.’s DXP introduces one approach in which the XP techniques can be utilized in more than one location. The problem of significant communication between team members, programmers and also the client is partially addressed in DXP, but it is limited to the use of email, teleconferencing and video conferencing.

Distributing XP development with Kircher et al.’s DXP techniques requires that the XP practices of planning game, pair programming, continuous integration and on-site customer have co-located teams. We have found that with XR, these practices can be tailored to suit distributed teams.

The following lists each of these practices and highlights the distributed approaches that can be adopted. The XP planning game requires the development team to meet the customer so as to evaluate risk, weight and priority of customer needs (user stories). By maintaining an online planning repository, the planning game can be played by distributed teams, on an asynchronous mode if necessary. Strict planning is valuable for any project and the planning, once adapted to research (see next section), enables more frequent feedback.

The pair programming practice stating that all code is written by two programmers sitting side by side at the same machine is more challenging for distributed teams, along with the fact that it goes against researchers’ traditional experience. However, using the appropriate tools, it is possible to perform remote pair programming, even though it may be of less benefit than traditional XP pair programming. It is a requirement that pair programming be undertaken at critical code areas such as software sub-system integration.
The continuous integration practice states that there is no ‘integration phase’. Researchers developing a proof of concept hope to get rapid feedback from code to refine a first idea. Having a centralized code factory is critical in a distributed environment, but modern tools make this possible.

The requirement of an ‘on-site customer’ for a distributed environment might sound impractical but we believe it is a necessary component. The on-site customer role can be performed by the principle investigator as he/she is ultimately the customer for the research. This enables the principle investigator to benefit from the same communication infrastructure as any member of the team.

XR emphasizes asynchronous communication to meet part of XP’s communication requirement. Rather than keeping all XP practices synchronous and then attempting to distribute them, the practices can be made asynchronous. This does not mean undertaking constant code reviews but instead increases the requirement for communication. Maintaining a planning repository facilitates access by team members at any time, and enables the members to explore decisions for the next iteration. This planning repository encourages the addition of material at any time and is not just limited to closed iterations.

5. EXTREME RESEARCHING

We have seen from the previous two sections that a distributed XP process for researching should contain the core principle of XP. In this section we examine the constituent components of XR and outline their importance in an applied research environment. A number of significant factors differ between XP applied in a standard industrial software development environment and that of a research environment. A number of adaptations are required to fulfil the requirements of an XP researching process; in Table I we identify the principle activities for the XR process and provide a description of these activities.

Our solution integrates spikes right into the process, not in the sense of simple coding spikes but rather spikes that have been clearly defined as real stories, weighted and planned for. We call these investigations controlled spikes. XR requires that these spikes be pair programmed, as they are at the same level as stories. Pair programming is not necessarily performed at all other stages of the project, as researching can involve specialist activities. The pair programming requirement is removed when the researcher so requests and this request is approved by the principle investigator.

For an applied research project, code is not always the core resource. Knowledge, understanding and the development of techniques are far more important than just the code itself. Dealing with these, the researchers focus on the metaphor rather than on the code they write. For this reason, a metaphor game is played so as to generate user stories. Each of these user stories is evaluated for its strategic value rather than a business value. Bellotti et al. [18] described in detail how such a metaphor game can be played. This metaphor continuously evolves while the code is written. This is precisely the requirement of researchers who are looking for something describing their ideas that reflects what they are implementing and enables them to get feedback from a partially working system (the proof of concept). We find that while the use of metaphors has been examined for general software development projects [19], it is a major improvement for research projects.

XR requires that a customer role be fulfilled, in a similar way to that of XP. In the research environment the principle investigators traditionally set forth the problem for the research; in essence they are the customers. It is for this reason that the principle investigator for the research is assigned the added role of customer. This has hidden benefits which only become visible during the execution of the project, primarily the fact that the principle investigator is also a member of the research team and so benefits from the communications of the team, stories and metaphors. A kind of brainstorming can take place before each release planning; during this, the entire team discuss and create stories and identify the spikes that need to be implemented. Following this, standard release planning takes place, when the customer chooses the stories and spikes of the highest priority. XR requires that object-oriented component-based modelling be used as a means to further communication and understanding between team members. The use of a modelling tool such as UML in an eXtreme process is novel, but experience in similar distributed teams [20] has shown that the use of such tools greatly improves the project.
Table I. The XR process.

<table>
<thead>
<tr>
<th>XR activity</th>
<th>Description of activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent integration</td>
<td>Frequent integration is facilitated by the use of an online, always on, code repository available for the distributed teams.</td>
</tr>
<tr>
<td>Remote pair programming</td>
<td>Remote pair programming enables the distributed pair to virtually sit together. This is accomplished via desktop sharing, videophone, Web cam etc., and is required in particular for controlled spikes.</td>
</tr>
<tr>
<td>On-site customer</td>
<td>The principle investigator fulfils the role of the on-site customer, who has final deciding powers. The customer’s requirements are supported by the stakeholders (the entire research team). This enables the planning game to keep focus on user stories.</td>
</tr>
<tr>
<td>Collective knowledge</td>
<td>The entire research team maintains a collective knowledge through constant reference to the online repository. This repository supports the collective ownership of the software and counteracts individual ownership.</td>
</tr>
<tr>
<td>Planning game</td>
<td>With distributed teams, the planning can be made a mix of synchronous and asynchronous communication: quickly discussed by phone, and then refined by each member virtually co-located. At the end of this period, a new phone conference declares the planning fixed. The planning game can be preceded by a metaphor game or brainstorming that focuses on prioritizing what needs to be implemented (refine the metaphor and define the user stories), while the planning game focuses on how much it will cost and what the priorities are. This is reinforced by the use of an online Web portal.</td>
</tr>
<tr>
<td>Metaphor</td>
<td>See planning game, metaphors are stored and updated through the use of the online repository.</td>
</tr>
<tr>
<td>40 h week on average</td>
<td>Researching suffers unduly from outside influences such as writing publications and grant funding. As a result, the number of hours per week can vary quite considerably.</td>
</tr>
<tr>
<td>Coding standards</td>
<td>The use of coding standards is necessary so as to harmonize the distributed teamwork; this together with the online repository improves understating.</td>
</tr>
<tr>
<td>Controlled software spikes</td>
<td>Through the use of controlled software spikes, divergence from the main research goal can be investigated and planned into the overall research project.</td>
</tr>
<tr>
<td>Testing</td>
<td>Unit testing is performed as with XP to insure the quality and validity of the research output.</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Refactoring occurs on the completion of an iteration when it is decided that the iteration is to be reused in the overall research project. Refactoring also occurs when a controlled software spike is reintroduced back into the research project.</td>
</tr>
<tr>
<td>Object-oriented component-based modelling</td>
<td>A key part of the XR process revolves around the use of component-based modelling; this facilitates the collective knowledge across the entire research team and allows for the controlled interaction of multidisciplinary research activities.</td>
</tr>
</tbody>
</table>

Table II. Tool support for the XR process.

<table>
<thead>
<tr>
<th>XR activity</th>
<th>Tool support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent integration</td>
<td>XPWeb, CVSWeb</td>
</tr>
<tr>
<td>Remote pair programming</td>
<td>XPWeb, iCom, Microsoft NetMeeting</td>
</tr>
<tr>
<td>On-site customer</td>
<td>XPWeb, iCom, Microsoft NetMeeting, Voice Communication</td>
</tr>
<tr>
<td>Collective knowledge game</td>
<td>XPWeb Planning</td>
</tr>
<tr>
<td>XPWeb Coding standards</td>
<td>XPWeb Metaphor</td>
</tr>
<tr>
<td>Controlled software spikes</td>
<td>XPWeb</td>
</tr>
<tr>
<td>Testing</td>
<td>XPWeb, xUnit</td>
</tr>
<tr>
<td>Object-oriented component-based modelling</td>
<td>XPWeb, rational rose</td>
</tr>
</tbody>
</table>

Finally, we find that a metaphor game played at the end of each iteration can be evaluated as very beneficial. All stakeholders must take part in this activity, as this is the primary point for feedback from the controlled spikes for the team.
6. ENABLERS FOR XR

XR requires two levels of communication. Initially, utilizing similar communication tools to that of DXP, but also adding to this suite, high bandwidth communication tools which aid virtual face-to-face mechanisms, such as PC to PC Web cam, video phone, video conferencing etc. Secondly, communication is facilitated by the use of the XPWeb Web portal and repository tool. This tool provides a virtual portal between multiple geographical locations and enables asynchronous team work, for a series, planning, pair programming, metaphor discussion and central repository (see Table II).

XP planning game tools are available [21,22] but these tools are limited in functionality, providing just planning tools. Similarly, many Web portal tools are available, Jakarta [19] for example, but they are not XP-oriented and so are difficult to integrate.

![Figure 1. Screenshot of automatically generated unit testing accessed with XPWeb](image)

To overcome these problems, it was decided to develop a combined repository and Web portal, XPWeb. XPWeb was designed particularly for distributed XP-orientated projects and provides the necessary repository requirements required by XR in particular. The central concept in XPWeb is to provide a tool set that facilities knowledge sharing in a distributed environment, together with a set of tools that allows for the development of proof of concepts.

XPWeb interfaces with the existing xUnit [23] testing framework (Figure 1), CVS, via CVSWeb for continuous integration of the project in a distributed environment [24]. In addition, enables browsing of object-oriented modelling though Rational Rose along with other features.

Some of these other features provide necessary support for XR: it has multi-language support, supports the metaphor and provides a metaphor linkage throughout the portal (Figure 2). Feedback for team members is improved by providing visualization of automatically generated reports based on previous iterations (Figure 3). In the applied research domain XPWeb facilitates our working in our multi-domain, multi-institutional and inter-discipline research projects.

XPWeb is open source [25] and plans for improving XPWeb include more integration with bug trackers, metaphor game implementation, a form of voting mechanism and a possible Eclipse Project plug-in.

7. OBSERVATIONS WHILE USING XR

Initially, we experimented with our adapted XP methodology, XR, with a research project examining pervasive computing. This project contained a number of inter-discipline research areas, all of which were interacting with each other. These disciplines included the following:

- Adaptive sensor networks: sensor technology has evolved from wired, simple transducers (e.g. for temperature) to complex sensors (e.g. chemical sensors, lab-on-a-chip devices, video and audio devices). These sensors have low-power consumption and have dynamic wireless networking capabilities; it is expected that these sensor networks will adapt to their environment, maximizing their usability and communicative efficiency.
- Content extraction from media: there is a continuing trend to develop mark-up languages (e.g. with XML, MPEG-7) for a variety of media, such as sensor data, text, video and audio, to allow content extraction and advanced semantic processing.

- Adaptive utilization: personalization research is moving to more advanced 'on-line' personalization and to 'situated' personalization, with systems adapting to individual/group needs, delivering relevant information in either the physical or virtual world based on deep profiles and sensor feedback.

This project was executed over 8 months with approximately 11,000-man hours being directly contributed to it to develop an advanced sensor test bed. Some small tasks related to the overall research program did not employ XR; as they were self-contained entities, we treated them like third-party components. The main goal of this prototype was to build a proof of concept interconnecting the above research fields. The main use case scenarios were focused on examining wide area service discovery, reliability and connectivity. Addressing these issues required building a team of researchers with a diverse skill set. Due to various human resource issues the team members were unable to be co-located in a single site for the entirety of the project. This resulting research project allowed us to evaluate the various facets and working assumptions described in our XR methodology.

One architectural component of the system was where the sensor network met wide area connectivity. Key issues for this subsystem were to examine new ideas and approaches where the three disciplines met.

The main components of a gateway are the following:

- Communication with sensor network: short-range wireless communication is used (Bluetooth, UWB, RF, IR, etc.). Provides functions like discovery of smart sensor nodes, generic methods for sending and receiving data to and from sensors, routing, etc.
- Gateway logic: controls gateway interfaces and data flow to and from the sensor network.
  It provides an abstraction level with API that describes the existing sensors and their characteristics; provides functions to uniformly access sensors regardless of their type, location or network topology, inject queries and tasks and collect replies.
- Communication with users: gateway communicates with users or other sensor networks over the Internet, wide area networks (GPRS, UMTS), satellite or some short-range communication technology. It is possible to build a hierarchy of gateways, i.e. to connect gateways described above to a backbone and then to provide a higher-level gateway that is used as a bridge to other networks and users.

In the above software subsystem we can see three research disciplines: embedded systems research optimizing short range radio communication for sensor networks; data modelling and representation research to effectively describe the sensor information; and finally research enabling the effective bridging of a 3G network to a sensor network. As we have seen from Section 5, XR allows for controlled spikes. One example of the effectiveness of this approach is when dealing with fundamental implementation issues such as platforms and simulators in a multi-decline environment. In this subsystem the hardware researchers were used to work with a specific simulation environment that only provided a C interface. The rest of the development utilized Java's advanced distribution techniques. In this instance we executed an XR spike with the objective of changing the hardware stack to that of an embedded Java one. The effective use of an XR spike in this case allowed us to investigate three alternatives and successfully change a key software component without affecting the key system stories.

The concept of XR metaphors was of critical importance when interconnecting the sensor information emanating from an embedded device across to a gateway node. This in turn adds semantic meaning to the information before offering the information onto wide area networks applications for further adaptation. One of the key findings from our initial use of XR was that, in the area of game playing, we effectively integrated the concepts of brainstorming and the customer roles when various parts of the subsystem were being verified.
8. DISCUSSION

Validation of any new process or methodology is always difficult, particularly during the formative years, partially due to the subjective nature of evaluation but primarily due to the limited experience in using the technique. Many approaches for examining the credibility, reusability and the efficiency of performing research have been discussed at length [26,27]. We have found that XR and XPWeb, when used in our in-house research projects, have reduced the project-overrun time on average by half². Currently we use a simple weighted metric based on customer feedback, patent filing and peer review publications. Our observations while using XR have suggested that we have yielded a significant increase in outputs (in the region
of 24%). As two research projects are never the same, it is difficult to exactly compare the productive gains while using XR.

Observations from adopting XR were in two critical areas. First, a major improvement in decreasing our lead time for outputs; previously multi-disciplinary projects delivered verifiable results only after system integration and testing. This decrease in lead time was of critical importance when it came to patent and standardization activities. The second benefit was removing the *big bang* approach of other previous methodology-driven projects, thus allowing effective feedback from project stakeholders throughout the project.

No matter how iterative one strives to make a methodology, normally an element of the classic waterfall process appears. This leads to the onset of *delayed decision-making* that fuels *big bangs* normally discovered in system integration stages of projects. Our use of controlled spikes endeavours to tackle this by tightly integrating risk into the process from an early stage. By forcing controlled branching as part of the overall process we create a set of *little bangs* that address risk and uncertainty from the outset of the project. With this strategy of being cognitive of the *big bangs* comes an overhead and an ownership issue. It is only when managed correctly through the use of tools such as XPWeb and the tight integration of controlled spikes into the development process that the problem can be successfully addressed.

9. CONCLUSION

The XR methodology was developed from the XP methodology, so as to support research (*applied research*) and development in a distributed environment. This environment involved changing human resources and rapid prototyping of telecommunications innovative software research. These were direct problems, which have been addressed by XR. We have examined the existing techniques, which have been used to address these problems and have highlighted their drawbacks, with regard to software support, metaphor and customer roles.

XR is specifically designed for use in the research phase of a software cycle and has not been tested out side of this.

In this phase, XR has been shown to increase the measurable output of a project by a significant margin when compared with existing techniques (*measuring patents, publications and successful commercial product development*). XPWeb, the support tool for XR, has proven extremely successful and flexible, and has recently been released in a multi-lingual version (currently nine languages).

To further evaluate XR we are actively employing quantitative techniques [28] to verify our empirical finding. Future work includes evaluating the usefulness of XR for projects outside of the research phase and examining larger research projects (currently under way).

Acknowledgements

We would like to thank Ericsson System Expertise for releasing the initial research results, all the members of the Applied Research Lab for participating in this study and also Dr James Power of NUI, Maynooth for his constructive comments on this paper.

==============
‡ Currently XR has been used on three other research projects not described in this paper.