2016 IEEE 24th International Requirements Engineering Conference

Listens Learned (8 Lessons Learned Applying EARS)

Alistair Mavin
Rolls-Royce plc
PO Box 31, Derby, UK
alistair.mavin@rolls-royce.com

Philip Wilkinson
Rolls-Royce plc
UK
philip.wilkinson@rolls-royce.com

Sarah Gregory
Intel Corporation
USA
sarah.c.gregory@ieee.org

Eero Uusitalo
IntoWorks
Finland
eero.uusitalo@intoworks.fi

***Abstract*—The Easy Approach to Requirements Syntax (EARS) is an approach for authoring natural language requirements using a simple template with an underlying ruleset. EARS applies a series of keywords to denote separate clauses within a requirement. Application of the template produces natural language requirements in a small number of patterns. EARS has proved popular with practitioners for numerous reasons. It is lightweight, easy to learn and easy to use. It helps authors to write clear, simple requirements that are easy to read and easy to check for errors. In this paper, four experienced EARS practitioners reflect on their experiences of applying the approach in numerous projects in diverse domains over a six year period. The paper provides an overview of the types of project in which EARS was implemented and describes the deployment methods used. During the course of these projects, numerous lessons were learned. The EARS practitioners discussed the lessons to reduce bias and to ensure the generalizability of the results. These include eleven general lessons concerning requirements engineering, elicitation and natural language specification. The main contribution of the paper is the eight key EARS-specific lessons learned during these deployments. These lessons learned are generalized and will scale to any EARS deployment. The lessons learned will be beneficial to practitioners who wish to deploy EARS in their own projects.**

Keywords - authoring; definition; EARS; elicitation; natural language; lessons learned; pattern; rigour; rules; stakeholder; template.

I. INTRODUCTION

A. Natural Language Requirements

Black-box system requirements are often written by individuals who are not experts in requirements definition. Most commonly, system requirements are written in unstructured natural language (NL) [7]. However, unconstrained use of NL is inherently unsuitable for requirements definition for a number of reasons. Common problems with NL requirements include: untestability, inappropriate implementation, wordiness, duplication, omission, complexity, vagueness and ambiguity [3].

To overcome problems associated with NL, some experts advocate the use of other notations for the specification of requirements (summarized in [3]). However, use of these non-textual notations often requires complex translation of the source requirements, which can introduce further errors. Such translation of requirements can create a "language barrier" between developers and stakeholders due to unconscious communication of assumed context [5]. There is also a training overhead associated with the introduction of many notations. Most requirement authors are unlikely to seek excessive formality and complex training, and rarely do they want or require a software tool to help them write.

B. Easy Approach to Requirements Syntax

The Easy Approach to Requirements Syntax (EARS) was developed to address the problems inherent in natural language requirements [3]. EARS provides a mechanism with which to "gently constrain" the use of NL. The EARS notation has been adopted by numerous organizations in different sectors and countries. Practitioners find the approach beneficial due to the low training overhead and the quality and readability of the resultant requirements [2, 8]. EARS is described is more detail in several previous papers [3, 4, 5].

When applying the EARS template to a series of case studies, the following basic requirements patterns emerged:

1) Ubiquitous

A Ubiquitous requirement is continually active and takes the form ***The <system name> shall <system response>***

2) State-driven

A State-driven requirement is active while some pre-condition remains true and takes the form ***WHILE <pre-condition> the <system name> shall <system response>***

3) Event-driven

An Event-driven requirement is activated when a triggering event is detected at the system boundary and takes the form ***WHEN <trigger> the <system name> shall <system response>***

4) Option

An Option requirement is used for systems that include a particular feature and take the form ***WHERE <feature is included> the <system name> shall <system response>***

5) Unwanted behaviour

An Unwanted Behavior requirement defines the required system response to an unwanted external event and takes the form ***IF <trigger> THEN the <system name> shall <system response>***

6) Complex

The basic EARS patterns can be combined to build Complex requirements, for example ***WHILE <pre-condition> WHEN <trigger> the <system name> shall <system response>***.

2332-6441/16 $31.00 © 2016 IEEE
DOI 10.1109/RE.2016.38

RE 2016, Beijing, China
Industry Paper

C. Structure of the Paper

The paper is deliberately short, with the specific aims that it is easy to read and provides simple practical advice. This is in keeping with the original intention of EARS. Section II of the paper briefly describes the domains in which EARS has been deployed, including the background of the organizations, the project contexts and the methods used to deploy EARS. Section III describes the general lessons learned for requirements engineering, elicitation and natural language. Section IV describes the EARS-specific lessons learned. Section V describes future work and a summary is provided in Section VI.

II. Domains and Methods Deployed

This paper includes the insights of four experienced EARS practitioners who have deployed EARS in a diverse set of domains over a six year period. The organizations, domains and methods of EARS deployment are described briefly below. The section also includes a brief discussion of the methods used.

A. Rolls-Royce PLC

1) Background

Rolls-Royce PLC is a global company providing integrated power and propulsion solutions in aerospace, marine, energy and other domains. Rolls-Royce employs over 40,000 people in more than 50 countries and has customers in over 150 countries. Many projects involve teams that are geographically distributed. The corporate language is English, although for many employees this is not their first language. EARS was originally developed whilst analyzing the requirements contained in the European Space Agency's "Certification Specification for Engines" regulations [10]. EARS has been applied on numerous projects in different business units and countries across the organization.

2) Method of EARS Deploment

Within Rolls-Royce, EARS training is provided by the company requirements expert as part of a half-day interactive training session "An Introduction to Requirements Engineering". The training is delivered on demand by self-selection through the company's Learning and Development system, or directly on request to specific teams. Engineers are typically trained in groups of between 8 and 16 persons. The training includes a single page summary sheet providing guidance on the essentials of EARS. Post-training reviews, coaching and support are provided.

B. Intel Corporation

1) Background

Intel Corporation is a highly complex, globally-distributed multinational corporation employing over 100,000. Jobs range from silicon manufacturing to research and development in a broad range of diverse technical areas. Intel operates in 46 countries worldwide and individuals as well as teams commonly engage with colleagues who are located in different countries. All staff are expected to converse in English if someone who does not speak the local language is present. All technical specifications, including requirements specifications, are written in English. The broad diversity of languages spoken, as well as varying degrees of English fluency, creates ample opportunities for miscommunication and misinterpretation. Requirements engineering at Intel is a distributed function with many different people with various titles and job responsibilities also being tasked with specifying requirements for a project or program. A very small number of RE Subject Matter Experts provide training and coaching to employees and teams worldwide, but no corporate mandate exists to use a single method, practice or tool for requirements activities. Only one business unit has a full-time dedicated Senior RE Subject Matter Expert. Two other experts work in a centralized group and support personnel working to develop expertise in RE in additional business groups. They also work with teams on short-to-medium term engagements, mentoring authors and helping facilitate the adoption of good practices in RE.

2) Method of EARS Deployment

Originally, a six-hour facilitated training session and workshop introduced the EARS concepts and provided an opportunity for teams to adapt requirements with a coach present [2]. Authors worked on specifications for two weeks, with coaching available on request. Based on these results and overall team satisfaction with the EARS patterns, EARS was incorporated into a one-day seminar, "Requirements Specification", which is taught on demand to teams, as well as through open enrollment Intel University sessions. Any employee of Intel Corporation, regardless of role, is eligible to attend this training. It is common for the majority of participants in any given class session to be employees in roles other than that of requirements engineer or author. Six years after the inclusion of EARS in the corporate RE curriculum, approximately 8500 people have received basic training in the use of the EARS patterns through "Requirements Specification" course attendance.

C. Civil Nuclear Power

1) Background

Nuclear power in Finland is controlled by law and decrees. For the purposes of practical interpretation, the Finnish Radiation and Nuclear Safety Authority, STUK, has developed the YVL guides, which specify safety requirements on the basis of law and decrees. The role of YVL is to provide detail and guidance on how to interpret the law in practice. In principle, YVL must remain neutral to different design solutions and is therefore a set of requirements documents. However, in practice YVL assumes the presence of certain design elements such as specific instrumentation. As a result of these factors, YVL contains both purposeful vagueness due to generic nature of the documents and design and implementation details that are due to the specific nature of the application domain.

2) Method of EARS Deployment

The application of EARS to YVL B.1 was performed by a researcher who had no previous background in the nuclear energy field at the time of the experiment in 2011. EARS

was applied to two other documents in the nuclear energy domain as part of a preliminary study and to selected sections of the YVL B.1 draft 2. Following this exercise, the EARS approach was applied to the YVL B.1 draft 2 in its entirety. Alongside the YVL requirements, EARS interpretations of each requirement were documented. 87% of the requirements were successfully reworded into the EARS format. This work is reported in more detail in a previous paper [8].

D. Method discussion

The methods described above were developed and implemented independently by the individual EARS practitioners. These separate instances of action research each led to a number of lessons learned when applying the EARS approach. The initial lessons learned were discussed amongst the EARS practitioners, in order to generalize the results. The discussion between multiple practitioners served to reduce any bias in the outcomes of the work. The resulting lessons learned are therefore widely applicable.

III. GENERAL LESSONS LEARNED

During the six years of EARS deployments described above, the EARS practitioners gained numerous insights. Some of these are not specific to EARS, but are more general in nature. These general lessons learned apply to any natural language requirements notation, requirements elicitation and documentation in general, or more widely to requirements engineering as a whole. These general lessons learned are summarized in this section of the paper. Some of these general lessons (and some of the EARS-specific lessons described in section IV) have been reported previously [6].

A. Keep it simple

All of the EARS practitioners agree that most stakeholders prefer simple language. A specific and often-cited advantage of EARS is that it does not require a specialist tool or notation. This lesson can be generalized to suggest that when deploying *any* approach or method for the first time, it is best to keep things as simple as possible. To reflect this, it is recommended to use simple language when training and supporting those using a new method or approach.

B. Introduce new approaches organically

Self-selected attendees of EARS training generally began to use the approach immediately upon receiving the training. They tended to feel the benefits quickly and naturally encourage those around them to try EARS. This "word-of-mouth" acts as a powerful mechanism to draw in potential attendees of the training. Human beings can sometimes be resistant to change, which may hinder the introduction of new approaches. Allowing a new method or approach to grow naturally and organically through "pull" appears to overcome much of this natural resistance.

C. Consistency is an asset

The purpose of a requirements document is to define what the system must do. Effective requirements documents achieve this using the simplest, most easily understood language. If a requirement must be read several times and the meaning is still unclear, then it is a poorly written requirement. One key benefit of EARS is that it drives consistency into requirements. This makes each requirement more alike and therefore easier to understand, since humans are very good at pattern recognition. However, this consistency can make a requirements document rather repetitive and dull. Requirements documents are rarely read for pleasure; they are not written for the purposes of entertainment. Stakeholders should be encouraged to accept the "sameness" of requirements expressions and embrace this as an advantage. The benefits of consistency, clarity and rigor, supported by the natural pattern recognition inherent in human beings far outweigh the downside of repetition. This consistency can be achieved through EARS, but any form of consistency would also bring this benefit.

D. Write iteratively

In practice, requirements are almost always written iteratively. Authors may begin with a subject about which a requirement may be needed. This may only be a single word, for example for an aero engine, subjects could include *Noise*, *Vibration, Temperature, Weight* and *Reliability*. Authors will use these seeds to draft and then improve requirements. This can take several iterations as development proceeds and more information becomes available. It is important that requirement authors understand this need for iteration and do not expect to get every requirement right first time.

E. Use requirement notations to introduce other techniques

As a new notation such as EARS is introduced and embraced by stakeholders, it can be used as a means to introduce other techniques and approaches that can aid the requirements engineering efforts. For example, scenarios can be used to elicit requirements [1]. Using everyday language and incrementally adding details can introduce such concepts to enhance the knowledge, skills and capabilities of the stakeholders. For instance, stakeholders could be invited to *tell a story*, which can then be presented as a *bubble chart*, then as an informal activity diagram. A similar approach can be used to introduce state models, truth tables and other useful techniques.

F. Is it really a requirement?

If a statement does not fit into the new notation, perhaps it is not a requirement at all. It could for example be an information statement or the rationale for another requirement. It could be something that the system has to achieve, but is too imprecise to be expressed as a requirement. Where this is the case, it could be that the statement needs to be written as a goal (or *aspiration*, or *target*). In such cases requirements would also need to be written that trace to the goal.

G. Should a requirement be split?

Many draft requirements will need to be broken down into more than one requirement to cover different states or events. This is quite a normal part of the development of a set of requirements (and is in effect part of the "Write iteratively" lesson above). One common indicator of a compound requirement that needs to be split is the presence of the word "or". If a requirement includes the word "or", it suggests that the required system behavior is not yet understood. Either the conditions that initiate a requirement or the system response need to be "unpacked" to remove the "or". This will usually result in two or more requirements.

Previous work [9] has shown that actors may, for certain types of system, become a significant source of confusion. Where a system has numerous actors, their relationship to the system, and their access rights can create some quite subtle requirements. In the requirement: *"When commanded, the control system shall select ignition"*, it is unclear who can initiate the command: the flight crew, the maintenance engineer, or both? If there are few actors, and their roles are well known, it may be easy to identify the implicit requirements.

However, in some case there may be a number of different actors who require access to a system and whose roles may not be immediately obvious. For instance, a security system for a business premises may contain a number of legitimate actors. Who should be authorized to deactivate the system? Should it just be the business proprietor? What about the building security staff and police? If they do all have deactivation authority, should they all have unique 'keys' whose use can be recorded? Explicit identification of the actors and their roles is essential if their relationships to the system are to be captured in clear requirements.

H. Consider wanted, then unwanted behaviours

It is advisable to elicit and draft requirements in two cycles. Firstly, consider how the system is required to behave when users and interacting systems behave "normally" (sometimes called "sunny-day behavior"). For authors using EARS, this will typically yield requirements in the *Ubiquitous*, *State-driven* and *Event-driven* patterns. A second cycle of elicitation can be used to determine what the system must do in response to any unexpected or unwanted behavior of users and interacting systems. This will produce requirements in the *Unwanted behavior* pattern.

Stakeholders frequently focus on successful or positive behavior. While understandable, they often overlook behavior that is thought be negative or bad. An effective way to counter such optimism is to ask series of "what if" questions. Questioning of this type frequently uncovers a raft of missing requirements and misplaced assumptions, which will yield additional requirements. For authors using the EARS notation, such requirements are explicitly denoted by the *If-Then* keywords of the *Unwanted behavior* pattern.

I. Use models to uncover missing requirements

Many requirements are still written in NL [7]. While NL can be beneficial, the use of various models to check for requirements coverage and consistency is essential in complex systems. Models such as scenarios and state transition diagrams can be an effective way to group related requirements into logical chunks of functionality. Both models can be used during requirements elicitation, as the basis for structuring a requirements document and during system verification. Scenarios are also especially useful in user-centric systems.

Whenever a model is used to provide extra coverage, it may be prudent to consider whether the models should be "hidden" from the stakeholders. Some stakeholders may be comfortable with various models, whilst others may not. If inappropriate models are used with stakeholders, this can be counter-productive.

J. Phase of operation

In many systems, functions operate over numerous phases of the operational profile or life-cycle of a product. For example, an engine thrust reverser should only deploy on the ground. Similarly, repairs and the testing of their effectiveness should only occur in the maintenance phase.

While such requirements may be obvious to subject matter experts, other requirements exist that are more subtle and less obvious. An engineer may misread the intent of a requirement and imbue it with the incorrect scope of operation. Consider the example of aero engine thrust control. Thrust must be controlled over the whole flight envelope, from *taxi* and *take-off* to *approach* and *landing*. Each phase may have specific thrust requirements that must be adhered to, which will generate a number of requirements.

K. Natural Language is not always the best notation

Not all requirements are best expressed in NL. Some requirements are better expressed in other formats, such as truth tables, graphics, mathematical formulas or Boolean algebra. It can actually be counter-productive to write these requirements using NL. Some diversity of requirements expression is actually desirable. A document may include 95% NL requirements, but the other 5% should be consciously documented using other more suitable notations. Some requirements are self-evidently suited to other notations. In aerospace this would include mathematical formulas used to calculate thrust, or flight envelopes, which should be specified using a graph. Non textual requirements should have a textual requirement that in effect points to the "object" that contains the pertinent information such as a table, graph or formula.

Where a draft NL requirement has too many clauses or conditions, this would suggest that the requirement should be expressed using another notation such as a truth table.

IV. EARS-SPECIFIC LESSONS LEARNED

The general lessons described above were identified as being important in the context of EARS deployment, but could also be applied more widely. More significantly, a number of important lessons were identified that are more specifically concerned with the application of EARS. The eight key lessons for EARS deployment are described in this section.

A. Training should be short

The simplicity of the EARS approach means that it can be learned quickly and people will immediately write better requirements. All of the experienced EARS practitioners recommend short training sessions. In general, EARS training is delivered as part of a half-day requirements engineering course [2, 8]. Small groups or individuals can even be instructed on how to use EARS in around one hour, provided they have experience in writing requirements and adequate support is given. To complement this short effective training, the EARS practitioners also recommend the use of a simple summary sheet. This can be used by requirements authors as an easy reference guide.

B. Use with or without a tool

One of the advantages of EARS is that it is a lightweight approach that does not rely on a specialist notation or any tool support. EARS can be deployed "on the back of a cigarette packet", in a simple application such Word or Excel, or with dedicated tool support. As mentioned in one of the general lessons, the EARS practitioners' experience shows that stakeholders tend to avoid specialist tools. EARS can be used effectively without a tool, which appears to be one of the reasons it is popular with practitioners.

C. Coaching to embed learning

Whilst EARS is simple and easy to learn, to write consistently good EARS-compliant requirements requires practice. To help authors embed the learning, coaching and support is highly recommended. There is an old Italian proverb which asserts that "Making wine is simple, but not easy". This could also be applied to using the EARS approach. The requirements that result from using EARS template are simple and easy to read. However, there can be a lot of effort required to reach this concise statement of need, which often requires support. To be able to write a clear and concise requirement, the author must first understand what the system must achieve, which may be difficult. This simplicity has the advantage that authors cannot hide behind ambiguous statements of what the system must do. Indeed, the simplicity of EARS requirements makes it easier to identify errors in the requirements.

As a part of coaching, early reviews help authors and in particular less experienced authors. Novices in the use of EARS are encouraged to write a small number of requirements (perhaps 10-20) which are reviewed by an experienced EARS practitioner. With the review comments in mind, authors can then write a further small set of requirements. This approach removes systematic errors, so that when a full document is produced it will contain less requirements errors. In turn this makes document review easier and reduces the need for unnecessary rework. An additional benefit is that the experienced practitioners spend less time reviewing.

D. Challenge the EARS Pattern

A key lesson is that authors should challenge their own requirements. This can be achieved by an individual or better still by encouraging colleagues to review draft requirements. The reviews should consider whether each requirement is using the correct EARS pattern. In particular, requirements that are written as Ubiquitous should be questioned. In practice, most requirements that at first appear to be *Ubiquitous* are likely to be either *State-driven*, *Event-driven* or *Unwanted Behavior* requirements.

It is possible to define essentially the same behavior using EARS states or events, since an event can cause a state transition. However, it is generally quite easy to determine which pattern is more appropriate. Similarly, practitioners sometimes initially write an *Unwanted Behavior* as an *Event-driven* requirement, but will usually recognize that the requirement relates to a failure or unwanted stimulus.

E. Are the EARS clauses necessary and sufficient?

Many functions require a specific set of clauses before they begin. Capturing such clauses with a requirement is an essential activity. It is important to determine which individual clauses are necessary for a function to begin, and which groups of clauses are sufficient for the function to begin. Usually, it is easy to identify the necessary clauses. Sufficiency, however, can be harder to determine, especially where numerous clauses exist. While such sifting of clauses, whether event, state or unwanted behavior, may be a burden, it does encourage requirements authors to question what is necessary and sufficient to for a system response.

The structure of an EARS requirement tends to help requirements decomposition. The clauses of a system requirement will often suggest the need for an element of functionality that has to be provided at a sub-system level. For example, where a system requirement is triggered by a change in temperature, there must be some means of detecting that temperature change. This will yield a derived requirement at sub-system or component level to measure temperature. In turn, this decomposition aids traceability between requirements at different levels in the requirements structure.

F. Non-functional requirements

EARS has been used in a limited capacity by some of the authors for non-functional requirements (NFR)[1]. Within Rolls-Royce (in both the Aerospace and Marine communities), the EARS patterns have been applied to a number of NFRs. Typical NFRs represented within these two communities fall into five common groups, which are described here.

[1] Although the term 'non-functional requirements' is widely used within RE circles, its usage is by no means universal. The natural language meaning of "non-functional" is *"does not work,"* which can result in confusion for authors who are not RE specialists themselves. Within Intel, the term is not in general use, with materials instead describing *Quality and Performance Requirements* and *Constraints*. While the latter are specified with EARS and are usually (if not always) Ubiquitous, the former require a different notation to ensure adequate specification.

Physical requirements such as weight, size and material type relate to some characteristic that the system should possess. As such they do not mandate a time-based system response. They have usually been written as a *Ubiquitous* requirement, for example "*The dry engine shall weigh less that x kg."*

Performance requirements such as accuracy and range can be represented by a *Ubiquitous* requirement, for example "*The engine control system shall measure engine thrust to an accuracy of + or – x lb."*

However, the *Ubiquitous* requirement is not suitable for all types of performance requirement. Some, such as time response may require the use of an *Event-driven* requirement, for example "*When fuel flow has been measured, the engine control system shall transmit the fuel flow measurement to the aircraft within x ms."*

Capacity requirements such as storage, flow and spares are easily handled by a *Ubiquitous* requirement. As in the case of physical requirements, they represent a characteristic, and not an actual system response, for example "*The engine oil tank shall have a storage capacity of x mls."*

Maintainability requirements, despite their often vague nature, have been represented using a number of the EARS patterns. Maintainability requirements at different levels of decomposition have successfully been represented, helping to provide greater precision. Examples include: "*When interrogated by the maintenance engineer, the engine control system shall display all fault codes.", "The engine control system shall display fault codes in the format specified by x.", "If the starter motor can be repaired, then the starter motor shall be repairable by the tools in toolkit x."* and *"If the starter motor cannot be repaired, then the starter motor shall be removable within x minutes."*

Implementation requirements define how the system should be developed, in terms of processes, methods, or use of existing solutions. These requirements do not drive a real-time system response, but are intended to drive a development response. They are usually handled using a *Ubiquitous* requirement, or occasionally by the *Option* requirement. For example "*The engine control system shall be developed to ISO standard 47623, version 2."* and *"Where the engine control system contains software, the software shall be developed to DO178C."*

The use of EARS for NFRs has raised some interesting questions. Is it legitimate to use the *system response* aspect of the pattern for a non-real-time system response? Some may argue that it 'bends' the template in a manner that was never originally intended. To some extent this may be true, but two of the authors have experienced considerable success with this application of EARS. A third author differentiates between two different types of NFR. Some NFRs are Boolean (for example *"The engine control system shall be developed to ISO standard 47623, version 2"*) and these can be written in EARS. Other NFRs are typically measured on a scale (for example *"When the power button is pushed, the laptop shall boot quickly"*) and these are not easily expressed using EARS. Further work is required to investigate the benefits and shortcomings of the use of EARS for NFRs and whether other notations may provide better coverage for quality and performance requirements in particular.

G. Safety and reliability requirements

EARS has been successfully used to represent a wide range of both functional and non-functional requirements for safety and reliability properties. Requirements that relate to failure rates are usually expressed as a *Ubiquitous* requirement: *"The total engine failure rate shall not exceed a MTBF of x failures per hr.", "The engine control system failure rate for inadvertent opening of the fuel shut-off value shall not exceed x failures per hr.", "The engine control system failure rate for inability to close the fuel shut-off value shall not exceed x failures per hour."*

Requirements that contain a qualitative failure requirement can be expressed as a ubiquitous requirement: *"The engine control system shall not contain single failure modes that command an inadvertent thrust reverser door deployment."*

While many failure rate and qualitative failure requirements can be stated in a *Ubiquitous* form, it may be appropriate to use the *State-driven* requirement, for example *"While in-flight, the engine control system shall not contain dual failure modes that together command a thrust reverser door deployment."*

EARS has also been used extensively to represent a range of functional safety requirements. All of the EARS patterns, except for the *Option* requirement, have been used to represent functional safety requirements: "*While in engine start-up, the engine control system shall detect all electrical faults in the protection function.", "While in engine start-up, if turbine temperature exceeds x ^{o}C, then the engine control system shall abort the engine start.", "If the computed airspeed is unavailable, then the engine control system shall use modelled airspeed.", "If Engine Pressure Ratio is unavailable, then the engine control system shall use Low Pressure engine shaft speed as the measure of engine thrust."*

H. EARS helps authors of all abilities

Once trained and using EARS, authors will often begin to use the style and language of the template, automatically fitting emerging requirements into EARS patterns, in some case without even realizing it. Surprisingly quickly, this mechanism will improve the quality of first draft requirements for both novices and experienced requirements authors. Experienced authors will often revisit existing sets of requirements and rewrite those using EARS, which will tend to uncover previously unidentified errors.

Within Intel, strong demand for training including EARS has come from locations where English is not the primary language spoken. Additionally, site demographics are often such that employees speak their local language at work, switching to English only to communicate with colleagues in other countries. Limited or irregular use of a language often results in less fluency with that language, especially any complex grammatical constructions. Demand for RE training, specifically including EARS, is very high in those locations, with the limited number of courses that can be

scheduled often oversubscribed. An engineer in Shanghai reported that she found value in EARS in contexts well outside of requirements specification.

V. FUTURE WORK

As mentioned in section IVF of the paper, EARS has been used for NFRs by some experienced practitioners, but only to a limited degree. The authors have adopted different approaches to NFRs, in some cases using EARS only for functional requirements. During the writing of this paper, the authors have recognized the need to undertake additional work in this area. Further work will therefore be undertaken to investigate the limits of the EARS approach for the expression of NFRs.

VI. SUMMARY

This paper presents the lessons learned from action research in the deployment of EARS carried out by four experienced practitioners over a six year period. The lessons learned are gained from multiple projects in diverse domains, many involving globally distributed teams. Each individual EARS practitioner gathered many insights, which were discussed by all four participants in order to generalize the resulting lessons. This discussion served to reduce the risk of individual bias on the results.

The work generated eleven general lessons learned that are applicable to any NL requirements notation, requirements elicitation and documentation in general, or in the wider context of requirements engineering. The paper also includes eight EARS-specific lessons learned. The scale and diversity of the EARS deployments included in this study suggest that the lessons learned are scalable to the deployment of EARS in any domain.

REFERENCES

[1] Alexander, I.F. & Maiden, N.A.M. (eds), "Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle" Wiley, 2004.

[2] Gregory, S., "Easy EARS: Rapid Application of the Easy Approach to Requirements Syntax", Industry Short Paper, 19th International Requirements Engineering Conference (RE2011), IEEE, Trento, September 2011.

[3] Mavin, A., Wilkinson, P., Harwood, A. and Novak, M., "EARS (Easy Approach to Requirements Syntax)", Proceedings of 17th International Requirements Engineering Conference (RE2009), pp 317-322, IEEE, Atlanta, September 2009.

[4] Mavin, A. and Wilkinson, P., "BIG EARS (The Return of Easy Approach to Requirements Syntax)", Proceedings of 18th International Requirements Engineering Conference (RE2010), pp 277-282, IEEE, Sydney, September 2010.

[5] Mavin, A., "Listen, then use EARS", IEEE Software Magazine, March/April 2012, pp 33-34, IEEE, 2012.

[6] Mavin, A. "Applying Requirements Templates in Practice: Lessons Learned" in Rupp, C. & Die SOPHISTen "Requirements-Engineering und-Management", Carl Hanser Verlag, 2014.

[7] Pohl, K. and Rupp, C., "Requirements Engineering Fundamentals", Rocky Nook, 2011.

[8] Uusitalo, E., Raatikainen, M., Männistö, T. and Tommila, T., Structured natural language requirements in nuclear energy domain towards improving regulatory guidelines, in Requirements Engineering and Law (RELAW), 2011 Fourth International Workshop on Requirements Engineering and Law, IEEE, 2011. p. 67-73.

[9] Wilkinson, P. and Mavin, A., "The Early Discovery of Requirements for Safety Critical Systems" Advances in Risk and Reliability Technology Symposium (ARRTS), Loughborough, June 2015.

[10] "Certification Specification for Engines" (CS-E) Amendment 1, 18-20, European Aviation Safety Agency (EASA), 10th December 2007, http://easa.europa.eu/ws_prod/g/rg_certspecs.php