International Journal of Computers and Informatics (IJCI)
Vol. (2), No. (4)

IJCI

المجلة الدولية للحاسبات والمعلوماتية
الإصدار (2)، العدد (4)

August 2023

Software Requirement Engineering: Traceability Techniques and Tools

Tarek Mohamed Nour
Lecturer of Information Systems, University of Tabouk, Kingdom of Suadi Arabia
tarekmnour@yahoo.com

Noura Abd Elrahman Albaladi
Master of Information Systems, University of Tabouk, Kingdom of Suadi Arabia
albaladi_noura2@hotmail.com

Abstract

Requirement Traceability is one of the activities in managing requirements. It is important for software projects and is affecting the quality of software products. Requirement Traceability is a method to analyze the effect of changes among various software development lifecycle parts. Agile methodologies have been presented as an alternative to traditional software engineering methodologies. The transformation between traditional and agile methodologies is a hard task so the need for traceability grows. This paper introduces traceability research at the requirement engineering on the traceability literature published during the last years. It also investigates and discusses the requirements for traceability issues. It finally presents several requirement traceability techniques and tools to support traceability.

Keywords: Requirements Traceability, Agile Software System, Requirements Traceability Tools.

1- Introduction

This research has focused on requirements traceability, which aims to study how to describe and follow the life of a requirement, in both forward and backward

E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

directions [1][2]. Many researchers have participated in the area for the last two decades [3][4][5], to provide solutions in the form of methods, tools, and a better understanding of traceability needs and challenges. The purpose of this paper is to review the development in traceability research of requirement engineering (RE) and the tools used to enhance traceability [6].

The traceability of software artifacts is considered an important factor in supporting various activities in the development process of a software system. In general, the objective of traceability is to improve the quality of software systems. More specially, traceability of information can be used to support various activities such as the change impact analysis, software maintenance, and evolution, and the reuse of software artifacts by identifying and comparing the requirements of the new system with those of the existing system.

The goal of software traceability is to discover relationships between software artifacts to facilitate the efficient retrieval of relevant information, which is necessary for many software engineering tasks.

Implementing traceability helps in conducting important tasks, such as the evaluation of how changes in an element may impact other parts of the system. Software maintenance is one of the core goals of implementing traceability in a software system.

This paper is organized as follows: Section (2) presents the definition of requirement traceability along with its types. Section (3) presents the related works through three parts (a) review Traceability in traditional software, (b) review Traceability in Agile software systems, and(c) tools used for traceability. Section (4) concludes the paper and outlines future research directions to enhance RS traceability.

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

2- Requirement Traceability

Gotel and Finkelstein stated that "Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forward and backward directions within software development process (i.e., from requirements, design, implementation, to testing and maintenance)" [2].

Pinheiro and Goguen state that "Requirements traceability refers to the ability to define, capture and follow the traces left by requirements on other elements of the software development environment and the traces left by those elements on requirements" [7]. Wieringa divided traceability into two groups in 1995 Forward and Backward Traceability [8] [9].

Forward traceability is the ability to trace a requirement to components of a design or implementation. Backward traceability is the ability to trace a requirement to its source, i.e., to a person, institution, law, argument, etc. While Lindval and Sandah decomposed traceability into two groups Vertical and Horizontal traceability as in [10].

Vertical traceability is tracing dependent items within a model. Horizontal traceability is tracing correspondent items between different models.

Backward and forward traceability are like horizontal traceability. Most of the research on requirements traceability was concentrated on horizontal traceability because it is more accurate than vertical traceability.

Pinheiro divides traceability into two groups [9]:

- Inter-requirements traceability refers to the relationships between requirements. Inter-requirements traceability is important for requirements change and evaluation. For example, when extracting all requirements derived from a specific requirement or its chain for refinement.

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

- Extra-requirements traceability refers to the relationships between requirements and other artifacts.

Inter-requirements traceability is like vertical traceability. There are three features that should be covered by a traceability model [9]: First Definition: The definition is related to the specification of the traces and traceable objects. Second Production: The production is related to the capture of traces, usually by means of an explicit registration of the objects and their relationships. Third Extraction: The extraction is related to the actual process of tracing, i.e., the retrieval of registered traces [7].

3- Related Work

3-1 Traceability in Traditional Software Development Process

In this section, all the manual traceability techniques are mentioned such as cross reference, document, and structured centered. Traceability is very hard since the requirements keep on changing throughout the lifecycle of software. The basic techniques to handle this process are mentioned with their small description in the table given below.

Table1. Manual traceability techniques

Technique	Description
1. Cross-reference centered.	In this technique documents which keep online forms are supported automatically by linking ends of cross-reference hyper textually. For example: forms of explicit requirements such as (tagging, numbering, or indexing) [2].
2. Document centered.	In this technique, requirements can be traced by describing either all or part of the content of the project documentation.
3. Structure centered.	In this technique helps in completing the requirement traceability by restructuring the document in the form of a graph or network.

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

The above techniques provide early feedback from the customers by which the analyst team gets more time to respond to the changes in requirements. Another important benefit is that the verification process becomes easier to implement. The graph or network view of a particular project which makes the requirement as a graphical representation is easy to implement.

3-2 Traceability in Agile Information Systems

Agile methodology enables the organization to deliver quickly. Change quickly. Change often [18]. While agile techniques vary in practices and emphasis, they follow the same principles behind the agile manifesto [36]:

- Working software is delivered frequently (weeks rather than months).
- Working software is the principal measure of progress.
- Customer satisfaction with rapid, continuous delivery of useful software.
- Even late changes in requirements are welcomed.
- Close daily cooperation between business people and developers.
- Face-to-face conversation is the best form of communication.
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity.
- Self-organizing teams.
- Regular adaptation to changing circumstances.

Agile development methods have been designed to solve the problem of delivering high-quality software on time under constantly and rapidly changing requirements and business environments.

The initial problem was how to add traceability in agile methods such as Scrum and Extreme Programming (XP). In Scrum [20], [21], a single person in the role of a

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

product owner (PO) is responsible for requirements elicitation and requirements prioritization. Requirements in Scrum reside in a product backlog, which is a prioritized list of all work items imagined for the software, which also can include technical improvements. The work items in the product backlog are called backlog items. Only the product owner can add new items to the backlog. The product owner works with a development team of five to nine cross-functional software developers. The PO and the development team conduct requirements analysis, requirements specification, and requirements validation informally and in collaboration. At the beginning of each two to four-week development iteration, based on the development team's previous performance, the PO and the development team decide which backlog items are implemented during the iteration. At the end of the iteration, the system validation is done by the PO, who reviews the new system behavior. Extreme Programming (XP) [38] concentrates on software construction and there is little guidance for requirements engineering. The actors in XP are an on-site customer and a team of three to twenty developers. The main RE practice in XP is the planning game, which begins with on-site customer writing requirements. Thereafter, the on-site customer and developers decide which of the requirements are to be implemented during the following two-week development iteration. The implemented requirements are also validated by the on-site customer.

Next, we present a summary of traceability modeling:

Cleland-Huang et al. [14] propose a Traceability Information Model (TIM). It creates traces between acceptance tests and user stories. RT is constructed by inserting a cross-reference to one or more user stories into each acceptance test. When the test cases are executed and passed, RT links are automatically created between the source code and test cases.

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

Taromirad et al. [15] proposed Domain-Specific Requirements Traceability (DSML) Used to build a traceability scheme for a specific domain or project.

Badreddin et al. [16] proposed Requirement Oriented Modeling and Programming Language (ROMPL) that supports the automatic generation and maintenance of the RT links between requirements, models, and code.

Ratanotayanon et al. [17] used a tool, namely Zelda Work with an agile software development process that captures and maintains links between high-level information and source code.

Espinoza et al. [18] proposed a traceability meta-model (TmM) that Supports creating RT by allowing user-defined RT links, well-defined roles, and linkage rules and specifying what kinds of RT links can and must be created.

Cleland-Huang et al. [19] used the Traceability (JITT) tool Which Provides an interactive domain for retrieving relevant code. The tool returns a list of elected classes and shows the relation between the retrieved class and the user story.

Duraisamy et al. [20] used RTM as a two-dimensional array that shows items in rows and columns. It is a technique to create RT links of requirements between product backlog and sprint backlog where the information is retrieved by using keyword searching functionality.

3-3 Tools for Traceability Management

Innoslate[1] [27] is a requirements management tool that allows import of requirements from other tools. By underlying model-based database, these relationships are automatically generated into Hierarchy Charts, Traceability Spider Diagrams, or 3D Traceability Diagrams in which requirements can edit requirements, or new requirements are created directly in these diagrams.

[1] https://www.innoslate.com/requirements-management/

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

Trace Maintainer [27] [28] [29] uses a rule engine and modifier for maintenance of links. It accepts change events in the case tool and provides element properties to the rule engine to update the traceability relations. The disadvantage of Trace Maintainer is that it does not support different types of traceability links, so the modifier should be written for the integration of this tool with any specific case tool.

Doors from IBM [21] [26] [27]is a requirements management solution that allows changing the attributes of the traceability links, gives the possibility to produce different types of traceability links between the artifacts, and stores all the documents that can be excel sheets, word files, and other rough documents into a centralized database.

ADAMS Re-Trace [27] [30]: is a Traceability Recovery Tool that compares the links retrieved by Latent Semantic Indexing (LSI) Information Retrieval with the links traced by the software engineer. If there are any contradictions the links are highlighted and built within the Advanced Artefact Management System (ADAMS).

Trust Analyzer tool [27] [31] establishes the traceability between scenarios which can exist in the form of plain English or diagrams and the source code. When applying the scenarios the internal activities of the system can be observed and recorded.

Rational Requisite Pro[2][27] is another tool from IBM that helps the management of requirements and allows traceability of one requirement to another. It can also be integrated with Rational ClearQuest Test Manager which manages all the test cases.

TraceM [27] [32]is a framework for automating the management of traceability relationships. It provides registration, integration, evolution and querying services. Information Integration and Open Hypermedia services are utilized by TraceM for

[2] http://www-01.ibm.com/support /knowledgecenter/SSRTLW_7.5.5/com.ibm.xtools.reqpro.doc/topics/c_trace.html

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

Traceability Management. Open Hypermedia allows the storing of relationships separately from the artifacts. Information integration provides the services for creating and maintaining evolving relationships between artifacts. Confluence[3] is open and collaborative, helping you create, manage, and collaborate on anything from product launch plans to marketing campaigns. Find work easily with dedicated and organized spaces, connect across teams, and integrate seamlessly with the Atlassian suite or customize with apps from our Marketplace.

REquirements Tracing On target (RETRO) [27] [33] is a tool for tracing requirements. It uses Information Retrieval methods (Vector Space Retrieval, Latent semantic indexing, and Keyword word extraction methods). The IR method is executed using reweighted query vectors. The methods are continuously enhanced with user feedback processing.

CRADLE[4][27] is a requirements management tool that enables traceability of items. Requirements can be imported from various sources like Excel and Word files and the traceability among different items can be established using the tool. The traceability and coverage of information can be analyzed using tables, matrices, and graphical hierarchy diagrams.

Trustrace proposed by Ali, Nawazish [27] [34], is a traceability recovery approach between requirements and source code using data mining. Trustrace uses a combination of both the Information Retrieval Method and Data mining to establish traceability links between requirements and source code. Trustrace is found to have more precision when compared to tools that use only Information retrieval (IR) methods.

[3] https://www.capterra.com/p/136446/Confluence/
[4] https://www.threesl.com/cradle/index.php

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

End to End Software Traceability tool is proposed by Asuncion, Hazeline [27] [35]. The tool follows three tiers architecture. This is the only tool that deals with end-to-end traceability of artifacts and was implemented for an organization.

Many tools are designed to help companies to test and manage their projects (https://www.softwaretestinghelp.com/software-testing-trends).

4- Conclusion and Future Work

The main goal of this paper is to investigate what is meant by traceability and how to trace the requirement. The topic highlighted is the study of many techniques used by different researchers. This research also draws some attention to the required traceability tools that are used to manipulate it.

The idea of tracing requirements presented in this paper has many opportunities for further expansion and research. The researchers in the traceability community established a roadmap [22] [23], and identified several challenges for traceability, including the Grand Challenge [24] to achieve Ubiquitous Traceability. These challenges are summarized as goals in Table (2). Each goal represents a required quality of traceability and is transformed to a set of research topics (described in detail in [22] [23]).

Table2. A Goal-Oriented Perspective [22] [37]

Goal Identified	Goal Description
Goal 1: Purposed	Traceability fit for purpose and should support stakeholder needs. So, it must be defined clearly for systems engineering tasks.
Goal 2: Cost-effective	Develop techniques for computing the return on investment (ROI) of traceability in a project. Supporting the effect of traceability decisions at different stages of the system life cycle.
Goal3: Trusted	Develop techniques for evaluating the state of traceability in a project so all the stakeholders can depend on the provided traceability.
Goal 4: Configurable	Develop techniques for dynamically generating and maintaining trace links that are configured according to project needs.

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

Goal 5: Scalable	Varying types of artifacts can be traced. develop techniques for scaling up traceability and for supporting multi-grained traceability across a variety of artifact types.
Goal 6: Portable	Traceability is changed and reused across projects and organizations so policies, standards, and formats must be developed for change and integrate traceability information.
Goal 7: Valued	Develop supporting techniques that bridge the technical and business domains of a project, so the benefits of traceability are visible and accessible to all stakeholders.
Goal 8: Dataset	availability of traceability datasets and benchmarks
Goal 9: Applications	real-world applications of traceability

Today, we have over 16 datasets available for community use from our CoEST.org website. The researchers designed an online Research Directions forum at "CoEST.org." this website has over 16 datasets available for community use and provides downloadable, executable, baselined experiments, developed in Trace-Lab [22] [25] [37]. Trace Lab is a different experimental environment which allows researchers to reuse, reproduce, and/or modify previous experiments, compose new experiments from a combination of existing and user-defined components, use publicly available datasets, exchange components, and relatively evaluate results against previous benchmarks. They provide links to these experiments from the Research Directions forum to encourage evaluation and generate the experimental results [22] [25].

Trace Lab is designed to enable future traceability research, by facilitating collaboration and creativity between researchers, decreasing the startup costs and effort of new traceability research projects, and encouraging technology transfer. Trace Lab has been released via CoEST.org in the summer of 2012. In addition, by late 2012 Trace Lab's source code was released as open-source software, licensed under GPL. Trace Lab currently runs on Windows but is designed to port to Linux and Mac environments [25].

IJCI, VSRP Publishing, UK
https://ijci.vsrp.co.uk
E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

Industrial and governmental organizations still work with tasks that could be reduced by more ubiquitous use of traceability – and industrial practice wants more tools and techniques from the research community [37].

The researchers should share in the ongoing discussion, to keep the community informed of important advances and new challenges as they grow, and where possible use existing experimental baselines or create new ones for others to use. This collaboration works will assist in achieving the ubiquitous traceability goal [22] [25].

References

[1] O. Gotel & Anthony" An Analysis of the Requirements Traceability Problem", C. W. Finkelstein,1993.

[2] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," in Proceedings Of 1st International Conference on Requirement Engineering, 1994, pp. 94-101.

[3] G. Spanoudakis and A. Zisman, "Software traceability: a roadmap", in Handbook of Software Eng. and Knowledge Engineering, 2005.

[4] R. Torkar, et al., "Requirements traceability: a systematic literature review and industry case study", International Journal of Software Engineering and Knowledge Engineering, vol. 22, no. 3, pp. 1-49, 2012.

[5] O. Gotel, et al., "The Quest for Ubiquity: A Roadmap for Software and Systems Traceability Research", RE 2012, pp.71-80.

[6] M. Narmanli, "A Business Rule Approach to Requirements Traceability", Sept. 2010.

[7] F. Pinheiro and J. Goguen, "An Object-Oriented Tool for Tracing Requirements," IEEE Software, vol. 13, no. 2, pp. 52-64, March 1996.

[8] R.J. Wieringa, "An Introduction to Requirements Traceability," Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam, September 1995.

[9] Francisco A. C. Pinheiro, "Requirements Traceability" in Perspectives on software requirements, Jorge Horacio Doorn, Ed.: Springer, 2003, Ch. 5, pp. 91-113.

E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

[10] M. Lindval and K. Sandahl, "Practical Implications of Traceability," Software Practice and Experience, vol. 26, no. 10, pp. 1161-1180, 1996.
[11] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," IEEE Transactions in Software Engineering, vol. 27, no. 1, pp. 58-93, 2001.
[12] A. Ghazarian, 2008, "Traceability Patterns: An Approach to Requirement-Component Traceability in Agile Software Development", Proceedings of the 8th WSEAS International Conference on Applied Computer Science, pg.: 236-241.
[13] M. Jacobsson "Implementing Traceability in Agile Software Development", 2009-02-02.
[14] J. Cleland-Huang, O. Gotel, and A. Zisman, "Software and systems traceability". Springer,2012, vol. 2, no. 3.
[15] M. Taromirad and R. F. Paige, "Agile requirements traceability using domain-specific modelling languages," in Proceedings of the 2012 Extreme Modeling Workshop, 2012, pp.45-50.
[16] O. Badreddin, A. Sturm, and T. C. Lethbridge, "Requirement traceability: A model-based approach," in Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International. IEEE, 2014, pp. 87-91.
[17] S. Ratanotayanon, S. E. Sim, and R. Gallardo-Valencia, "Supporting program comprehension in agile with links to user stories," in Agile Conference, 2009. AGILE'09. IEEE, 2009, pp.26-32.
[18] B. Arbain, A. Firdaus, I. Ghani, W. Kadir, and W. M. Nasir, "Agile non-functional requirements (NFR) traceability metamodel," in Software Engineering Conference (MySEC),2014 8th Malaysian. IEEE, 2014, pp. 228-233.
[19] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," Computer, no. 6, pp. 27-35, 2007.
[20] G. Duraisamy and R. Atan," Requirement traceability matrix through documentation for scrum methodology." Journal of Theoretical & Applied Information Technology, vol. 52, no. 2, pp. 154-159, 2013.
[21] M. Omar and J. Dhar," A Systematic Literature review of traceability Practices for Managing Software Requirements", Journal of Engineering and Applied Science 12 (Special Issue 4), Medwell Journals 2017

E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

[22] J. Cleland-Huang, O. Gotel, P. Mäder, A. Zisman, and J. Huffman Hayes "Software Traceability: Trends and Future Directions, ICSE '14 Hyderabad, India Copyright 2014 ACM 978-1-4503-2865-4/14/05.

[23] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grunbacher, and G. Antoniol. The quest for ubiquity: A roadmap for software and systems traceability research. In 21st IEEE International Requirements Engineering Conference (RE), pages 71 -80, 2012.

[24] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grunbacher, A. Dekhtyar, G. Antoniol, and J. Maletic. The grand challenge of traceability (v1.0). In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, Software and Systems Traceability, pages 343{409. Springer, 2012.

[25] E. Keenan, A. Czauderna, G. Leach, J. Cleland-Huang, Y. Shin, E. Moritz, M. Gethers, D. Poshyvanyk, J. Maletic, J. Huffman Hayes, A. Dekhtyar, D. Manukian, S. Hossein, and D. Hearn. Trace lab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. In Tool Demo,34th International Conference on Software Engineering (ICSE), pages 1375-1378, 2012.

[26] M. Taromirad, and R. F. Paige. "Agile Requirements Traceability Using Domain-Specific Modelling Languages". Proceedings of the 2012 Extreme Modeling Workshop on –XM '12 (2012).

[27] Satish C J, Anand M, and Thendral Puyalnithi," A Review of Tools for Traceability Management in Software Projects ", International Journal for Research in Emerging Science and Technology, Volume-3, Issue-3, Mar-2016.

[28] Mäder, Patrick, Orlena Gotel, and Ilka Philippow."Enabling automated traceability maintenance through the upkeep of traceability relations." Model Driven Architecture-Foundations and Applications. Springer Berlin Heidelberg, 2009.

[29] Mäder, Patrick, et al. "trace Maintainer-Automated Traceability Maintenance." International Requirements Engineering, 2008. RE'08. 16th IEEE. IEEE, 2008.

[30] Lucia, Andrea D., et al. "Adams re-trace: A traceability recovery tool. "Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on. IEEE, 2005.

[31] Alexander E. "scenario-driven approach to traceability." Proceedingsof the 23rd international conference on Software engineering. IEEE Computer Society, 2001.

E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1

[32] Sherba, Susanne A., Kenneth M. Anderson, and Maha Faisal."A framework for mapping traceability relationships." Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering. 2003.

[33] Sundaram, Senthil Karthikeyan, et al. "Assessing Traceability of software engineering artifacts." Requirements engineering 15.3 (2010): 313-335.

[34] Ali, Nawazish, Yann-Gael Gueneuc, and Giuliano Antoniol. "Trustrace: Mining software repositories to improve the accuracy of requirement traceability links." Software Engineering, IEEE Transactions on 39.5 (2013):725-741.

[35] Asuncion, Hazeline U., Frédéric François, and Richard N. Taylor. "An end-to-end industrial software traceability tool." Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering. ACM, 2007.

[36] Appleton, B. ACME Blog: Traceability and TRUST-ability. http://bradapp.blogspot.com/2005/03/traceability-and-trust-ability.html (2005, Tuesday, 15 March). Accessed June 2011.

[37] G. Antonio, J. Cleland- Huang, J. Hayes, M. Vierhauser," Grand Challenges of Traceability 2017", Cornel University.

[38] K. Beck and C. Andres. Extreme programming explained: embrace change. Addison-Wesley, Boston, MA, USA, 2nd edition, 2004.

E-ISSN 2976-9361
https://doi.org/10.59992/IJCI.2023.v2n4p1