Advances in Project Management [1]

Making sense of life cycle wars: From model to reality [2]

By Prof Darren Dalcher
School of Management, University of Lancaster
United Kingdom

This paper reconnects with the life cycle concept in light of recent conversations within the project management space seeking to create a more definitive or prescriptive formulation of life cycles. It draws upon and reflects on similar discourse and sentiments expressed within the software development community over forty ago, as well as opening a further dialogue and positioning additional contributions in this area, thereby setting the scene for renewed dialogue around the role life cycles as a project shaping, structuring and organising device.

In life cycles we trust

The notion of the life cycle is essential to any project, undertaking or initiative which endeavours to determine, devise, develop, deliver or deploy new content, structure, deliverables, artefacts, or any other form of change. One could argue that anything that extends beyond business-as-usual and requires a blueprint for change, therefore merits a life cycle-driven representation. Indeed, Professor Peter Morris, resolves that '*the project development (life) cycle is the one thing that differentiates projects from not projects*' (Morris, 2013; p. 116).

'*The skillset focusing on the life-cycle of projects is critical to both understanding and practising sound management. The life-cycle represents a path from the origin to completion of a venture. Division into phases enables managers to control and direct the activities in a disciplined, orderly and methodical way that is responsive to changes, adaptations and complications. Phases group together directly related sequences and types of activities to facilitate visibility and control thus enabling the completion of the venture*' (Dalcher, 2002; p. 60-2).

The 7th edition of the APM Body of Knowledge, reminds managers that whilst there are many ways of structuring and organising project-work, one of the more important shaping decisions revolves around the choice of the approach and the associated life cycle that matches that

[1] *The PMWJ Advances in Project Management series includes articles by **Prof Darren Dalcher**, who is also the series editor of the Routledge books on new and emerging concepts in program and project management. Prof Dalcher is also the editor and author of multiple Routledge books. See Darren's background and qualifications at the end of this article.*

[2] How to cite this paper: Dalcher, D. (2023). Making sense of life cycle wars: From model to reality, Advances in Project Management Series, *PM World Journal*, Volume XII, Issue VIII, August.

philosophy. Indeed, life cycles are fundamental to management of project-work. Dalcher (2002; 2019) provides a detailed rationale for the use of life cycles in projects. Morris (2013) observes that the life cycle brings out clearly the nature and characteristics of the work in different stages, and can also show the management actions needed to control and direct the project as it evolves through its life cycle. A crucial skill is to move the project at the appropriate pace through its development cycle (ibid.), doing the right thing at the right time. Ultimately, the project life-cycle acts as an important management and governance tool focusing on the allocation of resources, the integration of activities, the support of timely decision making, the reduction of risk and the provision of control mechanisms (Dalcher, 2002).

Given the relative importance of such decisions, it is hardly surprising that the choice of life cycle, and the mapping of potential alternatives can engender controversy and prove to be an emotive issue in many different settings. The next section revisits such early discourse and the emerging disagreements around the use of life cycles within the software development community, that seems to mirror many current day conversations in the project management world.

What's in a life cycle: Learning from the software engineering discourse

Many ideas adopted by the project management community, particularly around life cycles, iterative and evolutionary development and delivery, maturity and capability models and agile modes of working originate within the software development, information systems and software engineering arenas. Such concepts and ideas have often been borrowed and reused with very limited consideration of the new context and setting and scant regard for the applicability and implications of importing them across fundamentally different and diverse domains. It is therefore instructive to reflect on some of the conversations and debates related to these concepts, especially when the information is hard to find, lost over time or inaccessible.

Towards the end of the 1970s there was a growing global dissatisfaction with the state of software development. A substantial body of anecdotal evidence seemed to indicate that despite significant levels of investment, software projects were delivered late, over budget and tended to produce systems that did not reflect real user needs or expectations resulting in significant amounts of rework to render the products satisfactory. The state of the industry engendered many discussions and disagreements. This section will highlight one such exchange that occurred over two issues of the journal *ACM SIGSOFT Software Engineering Notes* back in 1982 and consider the implications that it raises.

At that time, the concept of the life cycle was being advocated as the all-powerful way of controlling and addressing the problem with software development. However, G. R. Gladden issued a passionate plea; '*stop the life cycle, I want to get off*'. Gladden (1982) reflected that the notion of the life cycle was increasingly harmful to the software development profession,

noting that the various forms of lifecycles had sought to describe the software development process as iterative events within the main tasks of design, implementation, test, etc. creating a required series of iterations undertaken as a better understanding of the system was acquired. Consequently, it could be noted that the extended iterations worked together to extend project schedules, invalidate designs, alter test requirements, and infuriate customers (ibid.). Consequently, insiders still bemoaned the lateness, incompleteness and error-proneness of delivered software. Whilst Gladden offered some alternatives, such as focusing on objectives instead of requirements and favouring physical demonstrations rather than written specifications, he highlighted the impact of incomplete requirements on the resulting process and the downstream modifications and adjustments that must inevitably follow.

Meanwhile, Daniel McCracken and Michael Jackson were invited to attend a conference held at Georgia State University in 1980, focussed on systems analysis and design, where the discussion was structured around the steps of the lifecycle. This seemed to offer an interesting perspective. Reflecting on their experience McCracken & Jackson (1982) concurred that the concept of the lifecycle can be considered harmful and noted the stultifying effect of such organisation. They reported that whilst most attendees found the life cycle concept to be comfortable, or comforting; they also pointed out the wide range of variations in interpretation between their own variations and those used by others. Yet, many of the attendees were also hopeful that all the different variations could simply be mapped onto one another.

In summary, McCracken & Jackson adduced three groups of criticisms:

1. To contend that any life cycle scheme, even with variations, could be applied to all system development is either to fly in the face of reality or to assume a life cycle so rudimentary so as to be vacuous. The detailed and overly elaborated life cycles often harbour historical assumptions or constraints that bind future development.
2. The life cycle concept perpetuates past failures, whilst ignoring users as well as changing needs.
3. The life cycle concept rigidifies thinking, thereby discounting the possibility that systems would be responsive to change.

McCracken and Jackson recognised a plethora of potential scenarios for development, including ones that may emphasise prototyping approaches or that focus on end-user centred development. They noted that many of these variations are unlikely to be mapped onto a standardised life cycle that is not vacuous, and therefore asserted that the traditional life cycle concept as originally conceived was simply unsuited to the needs of the 1980s in developing systems.

In the subsequent issue of the same journal, Patrick Hall reacts to the frontal attacks on the life cycle, offering a succinct rejoinder – repositioning the life cycle concept as a general description of the stages through which a system travels from its cradle to its grave; a clear

necessity for all systems. Hall (1982) affirms that there cannot possibly be a standard life cycle to encompass all circumstances given the differences between systems, although it might often be possible to identify specific similarities. The clear advantage that comes from the difficult task of predicting the life cycle is positioned as knowing the direction of travel, so that we can measure progress and determine the ultimate arrival. In other words, the life cycle concept could be used to guide and control the process. However, Hall immediately recognises that "of course a single life cycle imposed upon all projects is ruinous. Each project must declare its own life-cycle at the start, and then be monitored against it" (ibid.). Nonetheless, whilst eschewing prescriptive conformity, some general similarities can be identified and encouraged. Hall therefore concludes by encouraging a multiplicity of perspectives and representations that will enable greater flexibility. Instead, Hall warns about pedantic defenders of particular methods or approaches who do not seek to tailor and adapt them to the specific context.

Life cycles are clearly useful; however, the combined view is that the life cycle notion defies uniformity of purpose, needs or use. Instead, it requires recognition of situated context, thereby implying the need for contingency. Just before concluding his lament, Hall (1982) reflects on the earlier contributions published in the previous issue of the journal. He observes that McCracken and Jackson appear to have suffered from too rigid a view of a life cycle foisted by those seeking to impose a formalised structure, thereby preventing useful conversation and reflection. He therefore notes that the misapplication, or an oversimplification, of an idea should reflect on the people proposing it and not on the idea itself. Hall also asserts that iterations remain essential, and therefore rather than seek idealised representations, it is important to lay-down a suitable project-specific life cycle representation, and to monitor actual achievement against that proposal.

A rational life cycle and why it should be faked

In 1986, David Lorge Parnas and Paul Clements published a seminal paper entitled '*a rational design process: How and why to fake it*'. The paper was informed by the generalised experience from the design of a significant improvement and update to the A-7E Avionics System; a complex real-time embedded system devised for a carrier-capable subsonic attack fighter with a great payload and a significant range, in the US. Their position recognises the truly messy nature of large and complex systems that defy ordered resolution.

According to Parnas and Clements (1986) choosing a selected life cycle process is an idealisation because (slightly summarised and paraphrased, as follows):

1. In most cases the customer does not know what they want;
2. Many factors will not emerge until progress into the implementation has been made;
3. Human beings are unable to cope with all the details at once;
4. Projects are subject to change for external reasons;
5. Human errors cannot be avoided;

6. Preconceived ideas often influence, misguide and mislead developers; and,
7. Economic considerations may lead to less-than-ideal decisions.

There is an overarching tendency to seek over-simplification and assume that situations are logical and ordered. However, idealised design and development processes may not work in practice, particularly in fuzzy settings. Nevertheless, Parnas and Clements propose an array of reasons as to why the ideal design process should be followed and restructured:

- as guidance for designers;
- design will be improved by trying to keep as close as possible to a 'rational' method;
- as a standard which can encourage the transfer of people, ideas and software whilst enabling better design reviews;
- as a project measurement metric (against the 'ideal' process); and
- as a standard which can facilitate regular reviews by outsiders

Parnas and Clements viewed design documentation as the main medium of design and therefore expected the final version to be accurate and rational, almost regardless of the steps required to obtain such clarity. This implies designing and developing the software systematically, and where this strategy was not possible, 'faking' the documentation to make it look as if it was produced following the 'ideal' process. The messiness of real life encountered during the development phases can thus be eliminated with the benefit of hindsight, simplifying future maintenance, evolution and sustainment through the added clarity. Parnas and Clements conclude that in complex real-life settings it is very hard, if not impossible, to design rationally, and even 'faking' a rational process is difficult; nonetheless, if this results in a product which could be understood, maintained and reused, even retrospectively, the method of rational design would be well worth using.

Let the life cycle dialogue begin…

Life cycles have proved to be an emotive issue generating strong views and opinions; this has often been presented from a partial or limited perspective. In the spirit of the life cycle dialogues, the guest commentary article by Robert Buttrick (2023) this month is offered in response to a commentary article published in the June edition of PMWJ (Smith, 2023). Robert Buttrick is the author of a number of influential books (see, Buttrick, 2019; 2020), as well as multiple national and international standards. The purpose of the commentary article is to identify additional context and insights into life cycles, with a particular emphasis on highlighting recognised published standards. A lot of thought has gone into positioning the various standards and the paper endeavours to underline and share some of the more recent thinking. Buttrick is also able to identify and draw on the converging consensus across the different international standards, thereby highlighting key areas of interest to the wider community.

The key point made about the need to tailor and adjust methods to suit a context and enable effective management chimes with the observations of Hall (1982). It is also the hallmark of any form of effective management that extends beyond received recipes. The thinking behind hybrid life cycles which mix, match and fuse different life cycle patterns reflects a similar need to find the right balance. It starts with the recognition that context matters, and there is no universally applicable one-size-fits-all life cycle. Hybrid life cycles enable a pragmatic mix of philosophies, fusing together elements from both predictive and adaptive perspectives, thereby achieving more of what Gladden, McCracken & Jackson and Hall were all striving for back in 1982. Rather than create a prescriptive new model, tailoring and hybrid methods allows for the integration of experimentation, fact finding and prototyping in uncertain areas, with more predictive approaches in better understood areas. The APM body of knowledge offers additional benefits of tailoring, blending, merging or mashing life cycle elements, allowing for greater efficiency and flexibility (Murray-Webster & Dalcher, 2019).

Returning to the software development arena, the concept of capability maturity models also emerged from the defence sector around 1986. The underlying idea was to create a basic model that would be able to characterise the organisational capability of potential suppliers to deliver adequate systems. This was predicated on the early work of Ron Radice and his colleagues at IBM on the stages in the life cycle and the identification of 11 attributes focusing on awareness, knowledge, skills and wisdom and integrated management systems (Radice et al., 1985a; 1985b), which were integrated by Watts Humphrey at the Software Engineering Institute into the process maturity framework in 1986 (Humphrey 1988; Paulk, 2009). The basic representation characterised the progression across maturity levels from the initial and most basic level to repeatable, defined, managed and ultimately to the optimising level.

Whilst in the managed level, an organisation utilised a well-defined process to ensure consistent implementation, the optimising level assumed a foundation for continuous improvement, recognising the actual historical evolution in organisations and their evolving understanding of context, task and achievement. In practice, this means the ability to shift from interpreting and replicating to tailoring, thereby recognising the need for adjustments, local interpretation and regular and continuous improvement (Ginsberg & Quinn, 1995). Over time, developing the capability for tailoring and customisation of the models, rather than interpretation and adherence to pre-established structures became the hallmark of effective utilisation and successful implementation to reflect type, size, ambition, domain, setting and other practical considerations and realities (Casey & Richardson, 2004; Dalcher, 2004; Armmbrust et al., 2008; Dalcher, 2008; Kalus & Khurmann, 2013; Wells et al., 2015). Higher levels of maturity reflect a progression from following guidance to more sophisticated tailoring to context. Indeed, process tailoring, customisation and contextual adjustments are now an assumed and expected part of good practice.

Moving forward and rethinking the role of life cycles

The software community, much like systems engineering and the project management professions, has deployed a diversity of different representations categorised as descriptive, prescriptive and normative life cycles and approaches. Each one comes with a different rationale, unique perspective and a clear purpose, which need to be acknowledged when we are comparing, summarising and characterising the different options (Benediktsson et al., 2006). This section will therefore explore a list of considerations and concerns related to the choice, appropriateness and use of alternative life cycles models before highlighting some of the challenges inherent in comparing and applying models.

Contingency: Contingency trumps prescription: Decisions and choices are situated in very specific circumstances: Whilst we may wish to standardise and regulate different alternative options, there is a clear need to recognise the diversity of settings, needs and conditions where different life cycle scenarios, specific situations and alternative contexts may need to be recognised and addressed (Dalcher, 2021). Diversity should be viewed as a strength rather than a constraint, particularly in change-rich settings.

Perspective: It is important to recognise that life cycles are utilised by different actors for specific purposes. Organisational life cycles reflect organisational temporalities (Pinto, 2022) and change and growth over time. Commercial development is often captured through product life cycles. Benefits may show better when we deploy extended life cycles, which can also account for environmental impacts, sustainability concerns, and societal outcomes (Murray-Webster & Dalcher, 2019). Some agile representations endeavour to integrate different perspectives including a project cycle, a delivery cycle, an iteration cycle and an integration cycle, recognising the development episode, the iteration required, the delivery period, and the full project; thereby allowing for the interplay of nested processes with varying lengths and purposes. Meanwhile, supplier organisations and commercial management will similarly endeavour to amalgamate additional steps and activities into their life cycles to encompass a wider set of concerns, interests and needs (Dalcher 2015; 2017). Other perspectives such as technical and managerial may also need to be integrated:

"*The technical life-cycle identifies the activities and events required to provide an effective technical solution in the most cost-efficient manner. The technical domain dictates the shape and phases included within this cycle but the main focus is on the production of a technical solution. Project management literature however, refers to a plethora of other life cycles, including project life-cycles, product life-cycles, organisational life-cycles, acquisition life-cycles, implementation life-cycles, budget cycles*." (Dalcher, 2002; p. 60-2)

If we consider the different stakeholders, concerns and interests in a project, we may well end up with a wider scope that also encompasses sponsor life cycles, contractor life cycle, bidder life cycle, supplier life cycles and client life cycles. Whilst they may have certain aspects, phases or activities in common, they will inevitably emphasise different facets, deliverables

and systems of interest, thus reflecting the unique rationale and purpose of each unique viewpoint and representation.

Life cycle as an abstraction

Ultimately, the life cycle is a proposed model, which offers an abstraction, a simplification, of a rather complex reality. In other words, the model presents a simplification of reality with much of the detail left out, allowing a focus on the relevant and essential aspects that are viewed as being of specific interest.

However, there a number of considerations that are worth emphasising and remembering when models are being utilised and compared:

I. **Incompleteness**: A model is incomplete and distorted by definition, ignoring significant proportions of reality.

II. **Simplification**: A model offers a partial and somewhat distorted representation of that reality.

III. **Relevance**: like any other abstraction, a model is a static snapshot taken from a certain perspective with a defined purpose or reason – whilst they do not come with a health warning, they should be examined with respect to the pertinent purpose, perspective or reasoning.

IV. **Decay**: All generalisations decay over time; this could apply to the content and/or the assumptions that they embody.

V. **Accuracy**: Since any model is an inaccurate representation of reality, or a crude simplification, it is critical to know the rationale and assumptions that underpin its creation before utilising it.

VI. **Limitations**: In order for a model to be usefully deployed, its limitations need to be explicitly recognised.

VII. **Truth value**: Models betray a discrete subjective representation relevant to the observer; rather than offer an objective and universal description, the basis for the observation is selective and purposeful and therefore is limited by definition.

"All models are deliberately simplified, by our choice of which degrees of freedom to use as handles on reality; and all models of the world beyond the reach of our immediate senses are fictions, free inventions of the human mind" (Gribbin, 1995).

Models are extremely useful instruments for dealing with a (portion of a rather) complex reality. The observation that '*all models are wrong, but some are useful*' is widely attributed to British statistician George Box. The implication therefore, is that we should focus on whether something is useful and applicable to our specific purpose and case, rather than seeking to prove the universal applicability of a theory or model to all potential settings.

Relevance and utility offer a useful basis for judging our models. Indeed, Historian Yuval Noah Harari posited that '*scientists generally agree that no theory is 100% correct. Thus, the real test of knowledge is not truth but utility*'. The limitations inherent in our 'informed' models can therefore be more powerful and more critical to knowledge and decision making, leading Weizenbaum (1976) to caution that rather than become enchanted by them, teachers must beware the naïve simple-mindedness, that can result in a closing of the mind to the incompleteness and the limitations of our models.

Choosing an ideal life cycle model

Life cycle wars begin when orthodoxy and dogma prevail and there is little recognition of assumptions, limitations and compromises. To conclude the discourse regarding life cycle models and their general applicability, it is useful to highlight the inevitable trade-offs that they entail (often re-invoking the same issues that featured in the software life cycle dialogue reported above):

The map or the territory: Confusing the map with the territory is a common trap when we begin to unquestionably believe our simplifications, abstractions and models.

Completeness versus Understandability: Dalcher (2018) invokes Bonini's paradox to highlight the contrast between a complex reality and a model that is used to explain some simplified feature or aspect of that terrain: Put simply, realistic models are too complex for easy handling, even after undergoing significant simplification; the more detail included, the more difficult the conceptualisation.

Contextual relevance: Conveys the original purpose of a model or representation, thereby relating to a unique viewpoint, or perspective; what is deemed to be of value should reflect, justify and explain the **context of intervention**.

Generalisability versus Usefulness: Very simple models that promise general applicability may need to be described at a low level of detail to assure wider generalisability, potentially trivialising their value whilst compromising the possibility of wider useful contribution: Could equally be described as **sufficient detail versus relevance**.

Prescription versus Expectation: The life cycle model may be applied prescriptively as instructed or espoused, or be partially constructed and informed by the *a priori* plan and intention for the project (Pinto, 2022): Alternatively, it may also be selected after the

contingency of a situation has been understood, and the different approaches for obtaining a desired outcome have been considered, thereby informing the strategic choice of the approach to shape the project or programme.

Fidelity versus Adaptation: Fidelity to an original plan or model, particularly for interventions, is often contrasted with the ability to adapt to reflect particular settings or populations and their realities, needs or preferences: This may also relate to the level of detail and completeness invested in a model.

Usefulness versus Coercive Enforcement: Covers the potential usefulness or utility of a model against its use as a means of enforcing compliance for the purposes of management, uniformity, governance or excessively emphasising the tasks of cataloguing, simplification, categorisation or differentiation.

Above all, it important to remember that life cycle models serve a purpose. They are essential simplifications and their adoption in the particular setting and context in which they were conceived carries significant implications. Each model and perspective encompass hidden assumptions, and constraints reflecting their unique perspective. Ultimately, in seeking to group, reduce, simplify or otherwise delimit a set of abstractions and simplifications, one may risk robbing and depriving the descriptions and models of any useful content, detail, value and contextual relevance.

References

Armbrust, O., Ebell, J., Hammerschall, U., Münch, J., & Thoma, D. (2008). Experiences and results from tailoring and deploying a large process standard in a company. *Software Process: Improvement and Practice*, *13*(4), 301-309.

Benediktsson, O., Dalcher, D., & Thorbergsson, H. (2006). Comparison of software development life cycles: a multiproject experiment. *IEE Proceedings-Software*, *153*(3), 87-101.

Buttrick, R. (2019). *The project workout: The ultimate guide to directing and managing business-led projects*. Abingdon: Routledge.

Buttrick, R. (2020). *The Programme and Portfolio Workout: Directing Business-led Programmes and Portfolios*. Abingdon: Routledge

Buttrick, R. (2023). Response to Project Life Cycles* Sophie's Choice. *PM World Journal, 12*(8), August. https://pmworldlibrary.net/wp-content/uploads/2023/08/pmwj132-Aug2023-Buttrick-response-to-project-life-cycles.pdf

Casey, V., & Richardson, I. (2004). A practical application of the IDEAL model. *Software Process: Improvement and Practice*, *9*(3), 123-132.

Dalcher, D. (2002). Life cycle design and management. *Project Management Pathways*. High Wycombe: Association for Project Management.

Dalcher, D. (2004). Incremental capability improvement: the new weapon in addressing the all-pervasive complexity. *Software Process: Improvement and Practice*, *9*(3), 121-122.

Dalcher, D. (2008). Managing software processes: do we need new approaches? *Software Process: Improvement and Practice*, *13*(5), 383-385.

Dalcher, D. (2015). Whose success is it anyway? Rethinking the role of suppliers in projects. *PM World Journal, 4*(5), May. https://pmworldlibrary.net/wp-content/uploads/2015/05/pmwj34-May2015-Dalcher-whose-success-is-it-anyway-Advances-Series-Article.pdf

Dalcher, D. (2017). Commercial management and projects, a long overdue match? *PM World Journal*, *6*(10), June. https://pmworldlibrary.net/wp-content/uploads/2017/10/pmwj63-Oct2017-Dalcher-commercial-management-and-projects-article.pdf

Dalcher, D. (2018). The map is not the territory: Musings on complexity, people and models. *PM World Journal*, *7*(3). https://pmworldlibrary.net/wp-content/uploads/2018/03/pmwj68-Mar2018-Dalcher-map-is-not-the-territory-series-article.pdf

Dalcher, D. (2019). Moving beyond project delivery: Reflecting on the life cycle concept as way for organising project work. *PM World Journal*, *8*(1), 1-15. https://pmworldlibrary.net/wp-content/uploads/2019/01/pmwj78-Jan2019-Dalcher-moving-beyond-project-delivery.pdf

Dalcher, D. (2021). The power and peril of common standards: Knowledge and standards in project-work. *PM World Journal*, *10*(11). https://pmworldlibrary.net/wp-content/uploads/2021/11/pmwj111-Nov2021-Dalcher-the-power-and-peril-of-common-standards.pdf

Ginsberg, M. P., & Quinn, L. H. (1995). *Process tailoring and the software capability maturity model*. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Gladden, G. R. (1982). Stop the life-cycle, I want to get off. *ACM SIGSOFT Software Engineering Notes*, *7*(2), 35-39.

Gribbin, J. R. (1995). Schrödinger's kittens and the search for reality: solving the quantum mysteries. London: Weidenfeld and Nicolson.

Hall, P. A. (1982). In defence of life cycles. *ACM SIGSOFT Software Engineering Notes*, *7*(3), 23.

Humphrey, W. S. (1988). Characterizing the software process: a maturity framework. *IEEE software*, *5*(2), 73-79.

Kalus, G., & Kuhrmann, M. (2013, May). Criteria for software process tailoring: a systematic review. In *Proceedings of the 2013 International Conference on Software and System Process* (pp. 171-180).

McCracken, D. D., & Jackson, M. A. (1982). Life cycle concept considered harmful. *ACM SIGSOFT Software Engineering Notes*, *7*(2), 29-32.

Morris, P. W. (2013). *Reconstructing project management*. Chichester: John Wiley & Sons.

Murray-Webster, R., & Dalcher, D. (2019). *APM body of knowledge* (7th ed). Princes Risborough: Association for Project Management.

Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE transactions on software engineering*, *12*(2), 251-257.

Paulk, M. C. (2009). A history of the capability maturity model for software. *ASQ Software Quality Professional*, *12*(1), 5-19.

Pinto, J. K. (2022). Avoiding the inflection point: Project management theory and research after 40 years. *International Journal of Project Management*, *40*(1) 4-8.

Radice, R. A., Harding, J. T., Munnis, P. E., & Phillips, R. W. (1985a). A programming process study. *IBM Systems Journal*, *24*(2), 91-101.

Radice, R. A., Roth, N. K., O'Hara, A. C., & Ciarfella, W. A. (1985b). A programming process architecture. *IBM systems journal*, *24*(2), 79-90.

Smith, K. F. (2023). Project Life Cycle * Sophie's Choice: What's in a Word? commentary, *PM World Journal, 12*(6), June. https://pmworldlibrary.net/wp-content/uploads/2023/06/pmwj130-Jun2023-Smith-project-life-cycle-sophies-choice-2.pdf

Weizenbaum, J. (1976). Computer power and human reason: From judgment to calculation. New York: W. H. Freeman.

Wells, H., Dalcher, D., & Smyth, H. (2015, January). The adoption of agile management practices in a traditional project environment: An IT/IS Case Study. In *2015 48th Hawaii international conference on system sciences* (pp. 4446-4453). Piscataway, NJ: IEEE.

About the Author

Darren Dalcher, PhD

Author, Professor, Series Editor
Director, National Centre for Project Management
Lancaster University Management School, UK

Darren Dalcher, Ph.D., AKC, HonFAPM, FRSA, FBCS, CITP, FCMI, SMIEEE, SFHEA, MINCOSE is Professor in Strategic Project Management at Lancaster University, and founder and Director of the National Centre for Project Management (NCPM) in the UK. He has been named by the Association for Project Management (APM) as one of the top 10 "movers and shapers" in project management and was voted Project Magazine's "Academic of the Year" for his contribution in "integrating and weaving academic work with practice". Following industrial and consultancy experience in managing IT projects, Professor Dalcher gained his PhD in Software Engineering from King's College, University of London.

Professor Dalcher has written over 300 papers and book chapters on project management and software engineering. He is Editor-in-Chief of *Journal of Software: Evolution and Process,* a leading international software engineering journal. He is the editor of the book series, *Advances in Project Management*, published by Routledge and of the companion series *Fundamentals of Project Management*. Heavily involved in a variety of research projects and subjects, Professor Dalcher has built a reputation as leader and innovator in the areas of practice-based education and reflection in project management. He works with many major industrial and commercial organisations and government bodies.

Darren is an Honorary Fellow of the APM, a Chartered Fellow of the British Computer Society, a Fellow of the Chartered Management Institute, and the Royal Society of Arts, a Senior Member of the Institute of Electrical and Electronic Engineers, a Senior Fellow of the Higher Education Academy and a Member of the Project Management Institute (PMI), the British Academy of Management and the International Council on Systems Engineering. He is a Chartered IT Practitioner. He sits on numerous senior research and professional boards, including The PMI Academic Insight Team, the CMI Academic Council and the APM Group Ethics and Standards Governance Board as well as the British Library Management Book of the Year Panel. He is the Academic Advisor, author and co-Editor of the highly influential 7th edition of the APM Body of Knowledge. Prof Dalcher is an academic advisor for the *PM World Journal*. He can be contacted at d.dalcher@lancaster.ac.uk.