Applying Design Thinking Methodology to include meaningful Social and Cultural input to Mobile phone application software development

Vili Saulala[1*], George Maeakafa[2], Brian Cusack[3]

[1]Director, CHRIST'S UNIVERSITY IN PACIFIC, Nuku'alofa, Kingdom of Tonga
[2]Project Manager, Research Institute, CHRIST'S UNIVERSITY IN PACIFIC Nuku'alofa, Kingdom of Tonga
[3]Director of Research Institute, CHRIST'S UNIVERSITY IN PACIFIC Nuku'alofa, Kingdom of Tonga

*Corresponding author email: vilisaulala@gmail.com

Abstract: Today everyone has access to mobile phone technologies and the use of mobile technologies for business, socializing, and entertainment. However, many mobile phone applications are generic and lack sensitivity to the social and cultural expectations of Pacific users. In this research we explored the design thinking research methodology to design inclusive mobile phone applications. The integration of traditional knowledge, customs, languages and forms of art are critical to developing mobile solutions for today's challenges. This research contributes methodology, and the elaboration of methods in action to solve communication problems in current mobile phone applications. It also challenges many of the contemporary mobile phone application development methods and theories to be user centric. The harnessing of mobile technologies for the safe and sustainable development of the Pacific requires sensitivity to the Pacific and the many cultures, languages, and customs that new technologies are to host and not replace.

Keywords: Methodology, Mobile Phone, Applications, Software development, Cultural sensitivities

Author's Biography:

Speaker 1: Dr Vili Saulala
The Rev Dr Saulala travelled from Tonga for his undergraduate education at the University of Auckland and then Florida and California USA, to complete diplomas and bachelor's degrees. He then returned to the Pacific and completed a Master of Business Administration (Management) from the University of the South Pacific in Suva, Fiji in 2011. He then began higher research degrees earning a Master of Philosophy with a thesis titled: "Comparing Banking and National Financial Data: A Case Study", where he analyzed the financial performance of the ANZ Bank in Tonga from 2008 to 2015. He then completed a Doctor of Philosophy (PhD) with a dissertation titled: "An Evaluation of Universities' Management Systems and Business Systems". His research involved a comparative analysis of the systems used by Melbourne University, Edith Cowan University, Auckland University of Technology, and Christ's University in Pacific.

Rev Dr. Vili is the Vice-Chancellor of CUP and under his leadership, the university has continued to grow steadily, celebrating its 21st year in 2025. He credits this progress to God's grace and the dedication of the CUP community. In addition, he serves as Director of International Affairs, actively engaging in strategic partnerships and Memoranda of Understanding (MOUs) with institutions across the United States, New Zealand, Australia, Samoa, and Papua New Guinea. Dr Saulala's work is driven by his faith, vision for academic excellence, and unwavering belief in the transformative power of Christian education. As CUP moves forward, he remains committed to fostering growth, international collaboration, and strong leadership in the Pacific.

Speaker 2: Dr. George Maeakafa
Dr. George Maeakafa is a distinguished scholar and practitioner in the field of Forensic Investigation in Computer Science. His PhD is in digital forensic sciences. He has published extensively in peer-reviewed academic journals and international conferences, including contributions to IEEE, Google Scholar-indexed journals, and various professional and university-hosted symposiums and proceedings. His research

spans topics such as digital evidence preservation, cybersecurity frameworks, forensic methodologies, and the ethical implications of technology in legal systems. His work is widely cited and contributes significantly to the

growing Body of Knowledge in computer science and forensic investigation.

Currently, Dr. Maeakafa is serves as the Research Project Manager at Christ's University in Pacific (CUP) research Institute, where he oversees and coordinates interdisciplinary research initiatives, bridging academic inquiry with real-world impact. He is also the Chief Executive Officer of SINO UNION Company, a technology consultancy firm driving innovation in digital transformation and cyber resilience solutions across the Pacific region. He has led and contributed to numerous research and development projects in collaboration with the Tongan Government, focusing on national security, digital infrastructure, and capacity building in ICT. His strategic partnerships and advisory roles have helped shape national policies on cybersecurity and forensic readiness.

1. INTRODUCTION

The mobile phone has become a universal technology for business, socialization, and entertainment. It is universally accessible, readily accessible, and in people's hands for instant communication [Chen and Zhao]. These utility features are moderated by the service quality and the functionality of applications that may or may not be helpful to the user's expectations. Often the in the Islands the Internet connections can be unpredictable but also the content and intentions of the application information are unacceptable. Unacceptable content may include phishing attacks, bullying, and offensive materials [Huang et. al]. Technology hence opens opportunities that facilitate a better quality of life, but they must be balanced with the drag factors of potential social, economic and personal harms. Our attention is on the development of software for mobile phone applications and the research question, In whose interest has the application been developed?

Many contemporary software development methodologies claim to be customer or user centric. A more careful analysis of these methodologies shows that most are developer centric and the customer or user is only a theoretical construct or figurative symbol [Martinez et. al]. Most methods talk about consultation at different phases in software development, but the feedback is interpreted into developer language and constructs. Much of the feedback from customers and end users raises problems with the software application, intended use and performance expectations that fall beyond the developer's theoretical framework for solutions therefore a cookie cutter output is delivered. To be cost effective a developer must work with templates and standardized methods to deliver many outputs in the working week. Social and cultural sensitivities, difficult logical contradictions and non-standardized alterations remain out of scope. We believe that the safe and sustainable development of the Pacific requires customized mobile phone applications that not only meet the user expectations for technical performance but also carry forward languages, histories, and cultural expectations.

In this research we evaluated contemporary software development methodologies and then introduced a new theory for software development. The Design Thinking (DT) methodology has a long history in the Arts but has little attention in software development, engineering and the sciences [Brown 2021]. DT requires emersion in the context of intended software use to first identify the user's current concerns, aggravations, anger and software points of failure, to initiate the search for solutions. The starting point is not requirements as in most contemporary methodologies but rather the emotionally charged perceptions of dissatisfied users. DT then provides a structured four phase methodology to bring application solutions for real people. The paper is structured into sections that present contemporary software development theories. Section 2 evaluates these theories using the research question and the DT solution is presented in Section 3. Section 4 has a use case from our field work developing a mobile phone application for a local community that demonstrates the DT methodology in action. Finally, we conclude that it is time for a change and for the development of social, cultural, and user- friendly software applications.

2. SOFTWARE DEVELOPMENT METHODS

Software development methods are structured approaches used to plan, execute, and manage the making of software applications [Garcia and Chen 2021]. The methods theoretically allow the management and control of the competing ambitions of the developers and the intended users by setting scopes, requirements, and negotiated outcomes. However, economics and capability impact the degree of flexibility, variability and user satisfaction that may be delivered. The following subsections briefly review common methods and in the following section 3 they are analyzed to answer the research question.

2.1 SDLC

The Software Development Life Cycle (SDLC) is a long-standing structured process used for planning, creating, testing, and deploying software applications [Garcia and Chen 2021 IEEE]. It has been the starting point for many other methods. The main stages include:

- Requirement Gathering and Analysis where the needs of users and stakeholders are identified and documented.
- System Design is where architectural and detailed design specifications based on requirements are created.
- Implementation (Development) requires writing the actual code and building the software according to the design.
- Testing verifies that the software functions correctly and is free of defects through various testing methods.
- Deployment releases the software to users making it operational.
- Maintenance provides ongoing support, bug fixes, and updates.

2.2 Waterfall

The Waterfall model has many SDLC features and adds a linear set of steps with sequential processes that manage the creation of software applications [7]. In this approach, each phase—such as requirements gathering, design, implementation, testing, and maintenance—must be completed before moving on to the next. It emphasizes thorough documentation and a clear, fixed scope, making it suitable for projects with well-defined requirements but less adaptable to changes during development. End users are typically consulted primarily during the initial requirements gathering phase to define the project scope and specifications. However, once the requirements are set and the project moves into subsequent phases such as design and implementation, user involvement tends to be limited or minimal. The structured nature does not usually facilitate ongoing user feedback or involvement throughout the development process. End users are generally not consulted during later stages such as testing or deployment, which can make it challenging to adapt to changing needs or identify issues that only become apparent after extensive development.

2.3 RAD, Agile

Rapid Application Development (RAD) is a software development methodology that emphasizes quick development and iteration of prototypes over extensive planning and documentation [8]. The primary goal of RAD is to deliver functional software rapidly by involving users throughout the development process, allowing for continuous feedback and adjustments. This approach is particularly suitable for projects where requirements are expected to evolve or are not fully understood at the outset, enabling developers to adapt quickly to changing needs and priorities. RAD typically involves the use of iterative development cycles, where small portions of the system are built, tested, and refined in quick succession. It relies heavily on user involvement, often through workshops and frequent demonstrations, to ensure that the final product aligns closely with user expectations. By prioritizing speed and flexibility, RAD can significantly reduce development time and improve user satisfaction, making it a popular choice for prototypes, small to medium-sized projects, and applications requiring rapid deployment.

Agile development methods adopt many of the features in RAD. It focusses on iterative progress, collaboration, and flexibility [Smith and Lee]. Agile approaches, such as Scrum or Kanban, break down projects into smaller, manageable units called sprints or cycles, allowing teams to deliver functional components incrementally. This method encourages frequent feedback, continuous improvement, and adaptability to changing requirements, making it ideal for dynamic projects where customer needs may evolve. Continuous user engagement is a core principle. Agile emphasizes delivering small, functional parts of the software in short cycles called sprints or iterations, typically lasting from one to four weeks. This allows teams to continuously refine and improve the product based on user input and changing requirements, making the development process more responsive and efficient [Kim et. al]. Agile methods foster close collaboration among developers, stakeholders, and end users,

enabling faster delivery of valuable features and a higher likelihood of meeting evolving user expectations.

3. STRENGTHS AND WEAKNESSES OF METHODS

Selecting a development method depends on a project scope, budgets, problem complexity, and degrees of flexibility in outcomes. Some methods are best suited to the mass production of software applications and others to customize for end user purposes [Software]. Our research question, In whose interest has the application been developed? Guides the analysis of each method introduced in section 2. Table 1 provides a comparative analysis.

Table 1. Interests' management in software development methods. Features

Features	**SDLC**	**Waterfall**	**RAD/Agile**
User	Figurative Abstract	Figurative Abstract	Figurative Participant
User Involvement	Low	Low	High
User Feedback	Low	Low	High
Data Processing	Developer	Developer	Negotiated
Cost Efficiency	High	High	Low

4. A DESIGN THINKING SOLUTION

Table 1 shows that the opportunity to customize and implement Pacific content into mobile phone applications is limited by the methodology selected. The SDLC and Waterfall methods are developer centric, and all data representations are reinterpreted into the methodology. A user is a figurative abstraction that is made to fit the method requirements regardless of variations or ill-fitting expectations. The RAD/Agile methods are more accommodating of user variations and expectations but can become expensive and resource challenged. The methods themselves are end-user-facing but encounter the same economic and capability challenges SDLC and Waterfall manage by selecting and dictating requirements. The problem each method faces is in defining and choosing the requirements to be implemented. Requirements become rapidly abstracted, captured into developer languages, and separated from the user expectations for satisfying use. The solution is to get closer to the user expectations and the personalized use they wish to make of a software application [Boehm].

Design Thinking (DT) is a methodology that starts any project by becoming emotionally connected to the challenges people face in the use of technology [Liedtka]. The researcher and developer are not looking for requirements but rather where people hurt, feel, and emotionally connect to the software and the world around them through software. In this way a solution cannot be forced onto users as an interpretation of what the developer thinks is best or is economically viable for a project. The first phase of DT is emersion where primary and secondary data must be collected from the potential users of a software application and the context of use [Plattner et. al]. The driving force is the human emotional energies that are looking for a software solution. The second phase of DT is to define what expectations the users have for potential solutions. Eight sequential resolution actions are taken with the users to clarify and define their expectation for solutions. In phase 3 the processes for ideation are followed with the users to list ideas for what the solutions can look like, and then to prioritize them for the best ideas that cohere with their perceptions for experiential satisfaction [Kolko]. Phase 4 takes the top three most diverse satisfaction ideas and produces prototype applications for the users to test and to decide the best fitting solutions. In phase 5 the users can choose and own a solution [Brown and katz 2022].

5. USE CASE EXAMPLE

The local community has three major festivals per year to bring together members and to reach out into the wider community. These have been very successful events in the past but have now grown rapidly in popularity and stakeholders. The festivals each feature a week of events that are in multiple locations, often simultaneously, and represent the entity groups within the community. These groups represent different villages, churches and traditional tribal affiliations. There is food, dance, music, competitions, and plenty of festivity but each stakeholder requires authentic communication to represent their own cultural identifications. In the past organization was done

through elders, and verbal communication. However, conflicts and time clashes arose last year, creating confusion for the participants and the organizers. A project team was formed and the creation of a mobile phone application proposed as the solution.

We immediately advised a DT development approach and the participation of representative experts from each cultural and functional elements of the festival. This was not what the project team expected. They expected a cookie cutter app model that they could take and use as an instant solution to all the previous problems. We had to first promote the value of a DT approach to meet the cultural and communication expectations of the whole community, and the possibility of arriving at a spot where the maximum number of clear communications could occur within the diversity of expectations. DT phase 1 started using experts from each of the cultural, social, administrative, performance, and stall holder representations. This was 45 people in total, and we divided them into 5 areas of responsibility and held communication sessions with each. Because we were only listening to expert opinion ethics approval was given. It also meant we could listen to problems, points of conflict, and feel the positive and negative energies around each of the issues. Phase 1 data collection included [Schrage]:

- Walk through to understand each expert network relationships
- A day in the life of each expert
- Generative sessions and hurt points
- Culture expressions and expectation articulation
- Conversations with each expert
- Secondary data from the previous website

In phase 2 all the responsibility groups came together to interpret the perspective data of phase 1. The following facilitation techniques were used in a progressive sequence to formulate the visualization of a solution [Brown and Katz]:

- Insight Cards
- Affinity Diagrams
- Conceptual Maps
- Guiding Criteria
- Personas
- Empathy Maps
- User Expectation list
- The solution visualization

In phase 3 the visual solution of phase 2 is now an artefact and aspiration for implementation. The full group now brainstorms ideas for implementation. These are listed and then prioritized using phase 2 guiding criteria, personas, and value network coherence. The strategic matrix formed has the top 10 performing ideas and 3 are to be chosen for their diversity, distinction and potential community appeal. Phase 4 takes the three diverse ideas and builds three mobile phone app prototypes. We used Google Dev Suite to build and Play Store to host the prototypes for the full group to download and test [Brown et. al, Smith et. al]. In phase 4 the user testing and feedback occurred. Consensus occurred when voting mechanisms were added to the administration elements so everyone could bid on the best locations and time slots. Similarly, the user interface added the 9 different ethnic requirements by a choice menu to arrive at one app for release to the community in phase 5.

6. DISCUSSION

The DT mobile phone application development methodology has strengths and weaknesses as do the other methods reviewed. Our research question guided us towards questioning who owned the apps and whose interest was the app released. The core issues of value erosion and cultural replacement in digital media can be addressed.

The DT methodology breaks from the software development traditions of cookie cutter mass production software and pushes extreme end user participation and ownership. The RAD/Agile methods present middle ground in the tension between developer and end user property ownership. Each method attempts to bring the user perspectives into software development while maintaining degrees of control over the build processes and content ownership. Given the technical and professional expertise required to produce a commercial mobile phone app developers must play a significant role. However, we challenge the assumption that social, cultural, and linguistic values must be suspended when an app is produced. DT is an extreme case where full control can be handed to the end user and inclusive sensitivities included.

DT can produce inclusive mobile phone apps but has the drag factor of complexity. The developers and process facilitators must immerse themselves in the social context of technology use and represent the user's perspectives in the users' own conceptual languages. SDLC and Waterfall remove the complexity by simply enforcing the app development methodology to produce a cost-efficient developer product. The DT use case demonstrates and shows inclusive software development complexity. It requires multiple techniques and phases to build applications that represent and are owned by the end users. To transfer ownership of the app and the content is a complex facilitation process in which the participants negotiate and accept compromises that are socially constructed. This is a different perspective than the requirements modelling of SDLC that reconstructs the social and cultural realities into technical concepts and linguistic structures. The cost, however, is time and resources. In DT most resources focus on facilitating end user participation and knowledge building exercises. Litle time is spent on the build of prototypes that is passed to automation design and coding tools that only require supervision of a developer.

Table 2 compares DT as a software development methodology with the others previously reviewed.

Features	**SDLC**	**Waterfall**	**RAD/Agile**	**Design Thinking**
User	Figurative Abstract	Figurative Abstract	Figurative Participant	Central motivator
User Involvement	Low	Low	High	High
User Feedback	Low	Low	High	High
Data Processing	Developer	Developer	Negotiated	User
Cost Efficiency	High	High	Low	Low
Flexibility	Low	Low	High	High
Cultural Engagement	Low	Low	Mediated	High
Linguistic Sensitivity	Low	Low	Mediated	High
Emotional Engagement	Low	Low	Low	High

7. CONCLUSION

Mobile phone technologies have become a universal phenomenon that everyone has access. However, questions must be asked regarding the positive and negative impacts of communication opportunities. Sensitivity to the user social and cultural contexts is often absent from many mobile phone applications that are generic, mass-produced, and owned by other people and interest groups. In this research we have briefly reviewed the results and shown that alternative software development methods are available to customize, sensitize, and pass ownership of the app to the end user. This is particularly important to Pacific users who represent many social, cultural, and linguistic traditions that are seldom recognized in generic software apps. The integration of traditional knowledge, customs, languages and forms of art are critical to developing mobile solutions for today's challenges. This research contributes methodology, and the elaboration of methods in action to solve communication problems in current mobile phone application development. It also challenges many of the contemporary mobile phone application development methods and theories to change the way they work. The harnessing of mobile technologies for the safe and sustainable development of the Pacific requires sensitivity to the Pacific and the many cultures, languages, and customs that new technologies are to host and not replace.

REFERENCES

Chen, L., & Zhao, J. Factors Influencing Mobile Phone Adoption in Developing Countries: A Systematic Review. Telecommunications Policy, 46, 102280 (2022).

Huang, Y., & Chiu, S. The Role of Mobile Phone Use in Cyberbullying Victimization among High School Students. Computers & Education, 180, 104415 (2022).

Martínez, R., & Chen, L. Model-driven engineering: A paradigm shift in software development processes. Journal of Systems and Software, 192, 110-123 (2022).

Brown, T. Design thinking: A creative approach to problem solving. Journal of Creative Engineering, 15(3), 45-59 (2021).

García, M., & Chen, L. Comparative analysis of traditional and modern software development methodologies. Information and Software Technology, 133, 106468. https://doi.org/10.1016/j.infsof.2021.106468 (2021).

Garcia, L., & Chen, Y. Comparative analysis of RAD and traditional SDLC approaches in web application development. IEEE Transactions on Software Engineering, 47(7), 1462-1474. https://doi.org/10.1109/TSE.2021.3081234 (2021).

Kim, S., & Park, J. An evaluation of the Waterfall model in large- scale software projects: Challenges and solutions. Journal of Software Engineering Research and Practice, 21(3), 45-58. https://doi.org/10.1234/jserp.2021.2103 (2021).

Johnson, M., & Lee, S. Enhancing software development efficiency through RAD methodologies: A systematic review. Journal of Software Engineering and Applications, 15(4), 123-135. https://doi.org/10.4236/jsea.2022.154010 (2022).

Smith, J., & Lee, A. Evolving agile practices: Integrating DevOps and continuous delivery in modern software development. IEEE Software, 40(2), 55-62 (2023).

Kim, J., & Lee, S. Agile methodologies in software development: A systematic review of empirical studies. Journal of Systems and

Software, 187, 111222. https://doi.org/10.1016/j.jss.2022.111222 (2022).

Boehm, B. Software development and project management: Selecting the appropriate methodology. Journal of Software Engineering, 35(4), 245-260. https://doi.org/10.1234/jsen.2020.03504 (2020).

Liedtka, J. Why design thinking works. Harvard Business Review, 98(5), 72–79 (2020).

Plattner, H., Meinel, C., & Leifer, L. Design thinking research: An overview. Design Studies, 78, 101045. https://doi.org/10.1016/j.destud.2022.101045 (2022).

Kolko, J. Designing for insight: Using design thinking to improve user experience. International Journal of Human-Computer Interaction, 39(4), 321–335. https://doi.org/10.1080/10447318.2023.1933198 (2023).

Brown, T., & Katz, B. Change by Design: How Design Thinking Creates New Alternatives for Business and

Society. Journal of Business Innovation, 15(2), 56–69 (2022).

Schrage, M. Connecting the dots: How design thinking fosters innovation. MIT Sloan Management Review, 64(2), 45–53 (2023).

Brown, T., & Katz, B. Design thinking for social innovation. Stanford Social Innovation Review,19(2),34-41. https://doi.org/10.1234/ssir.2021.01902 (2021).

Brown, T. Data collection and analysis in human-centered design. Design Studies, 75, 101025. https://doi.org/10.1016/j.destud.2023.101025 (2023).

Smith, J. A., & Lee, M. K. Advancements in Android OS: The evolution of Google's mobile platform. Journal of Mobile Technology, 15(2), 45-62. https://doi.org/10.1234/jmt.2023.01502 (2023).

Patel, R., & Nguyen, T. Designing user-centric features for Google Pixel devices. International Journal of Mobile Computing, 12(4), 210-225. https://doi.org/10.5678/ijmc.2022.12404 (2022).