DERON
INNOVATING THE FUTURE

10 STRATEGIES FOR MANAGING SOFTWARE DEVELOPERS

DOUGLAS ONYANGO - FOUNDER, DERON LIMITED

Phone: +256 776 716 138 Email: ondouglas @deronltd.com

ABOUT ME

Founder of Deron Limited, a Ugandan-based ICT and Innovations company that provides automation and digitization solutions for several clients. These include the British High Commission, UNDP, PACE, Malaria Consortium, Save The Children, and various Ugandan ministries such as Health, Agriculture, Gender, Labor, and Social Development.

Most notably, he is recently leading the development of the Parish Development Model Information System (PDMIS) for the Ministry of ICT and National Guidance.

3. DERON LIMITED

A local ICT and innovations company

DERON
INNOVATING THE FUTURE

DERON LIMITED is an ICT innovations and data management company that specializes in providing solutions that power businesses around East Africa.

We specialize in solutions for both government and private sector in finance, health, agriculture, social development, law enforcement and education.

KEY SECTORS:

HEALTH | AGRICULTURE | FINANCE | EDUCATION
SOCIAL DEVELOPMENT

SATISFIED CLIENTS

REPUBLIC OF UGANDA
MINISTRY OF FOREIGN AFFAIRS

British
High Commission

pace

REPUBLIC OF UGANDA
MINISTRY OF HEALTH

Save the
Children.

malaria
consortium

World Health
Organization
Country Office, Uganda

UNDP

10 STRATEGIES FOR MANAGING DEVELOPERS

1. Hire Slow:
2. Understand Your Developer
3. Learn Some Tech
4. Set up Standards/Rules/Guidelines
5. Clear Communication
6. Empower Your Developers
7. Build your team
8. Access to Your Code/IP
9. Eliminate Single Points of Failure
10. Fire Quick

1. HIRE SLOW

Why Hire Slow:
'Hire Slow’ allows for a thorough evaluation of a candidate’s technical abilities, work ethic, and cultural fit, increasing chances for a better candidate.

How to 'Hire Slow':
To implement 'Hire Slow,' use detailed job descriptions and a multi-stage hiring process. This includes technical assessments, behavioral interviews, and involving various team members in the evaluation. Reference checks are a must.

Note:
Aim for a thorough yet efficient process that makes informed decisions promptly, avoiding unnecessary delays in team growth and project progress.

2. Understand Your Developer

Why Understand Your Developer:
Knowing your developers' motivations and capabilities enables personalized management that enhances productivity and job satisfaction.

How to Understand Your Developer:
Engage in regular one-on-one meetings, conduct surveys, and foster an environment where feedback is welcomed and acted upon.

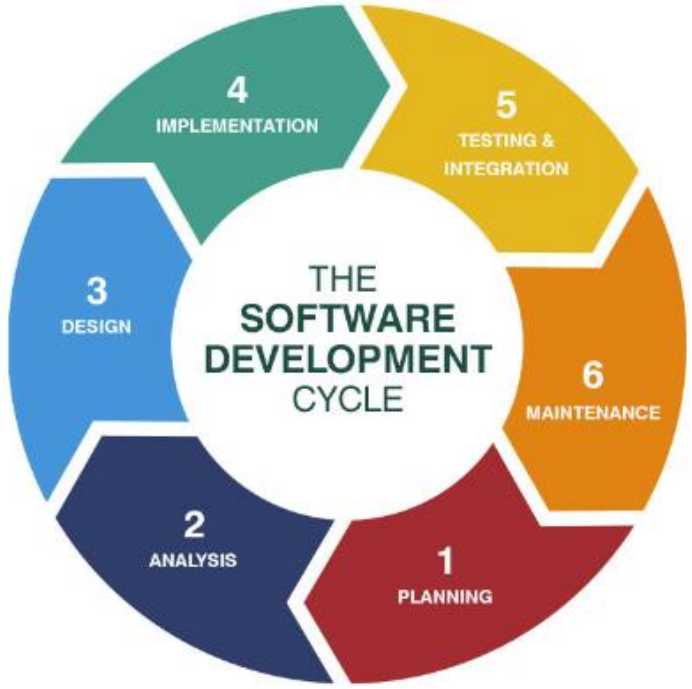
Note:
Balance understanding with privacy; maintain professional boundaries while showing genuine interest in your developers' well-being.

3. LEARN SOME TECH

Why Learn Some Tech: Basic technical knowledge helps you communicate effectively with your team and make informed decisions. In addition to basic knowledge on technology learn project management, business analysis, software testing and version control etc.

How to Learn Some Tech: Take introductory courses, attend workshops, and spend time with your development team to gain insights into their daily challenges.

Note: Avoid overstepping into technical micromanagement; use your knowledge to support your team, not to dictate

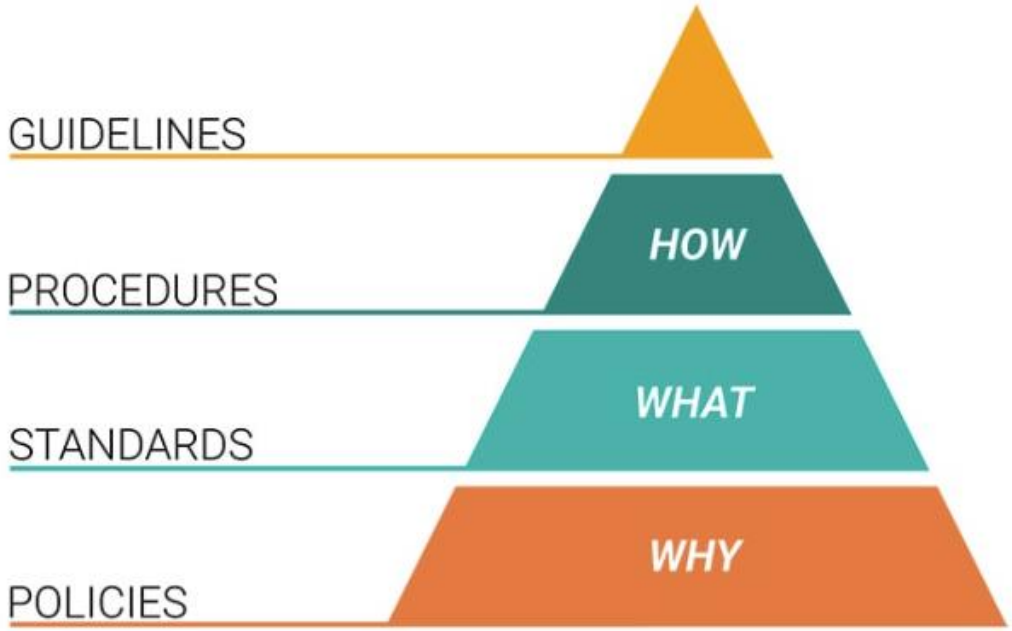


4. SET UP STANDARDS AND GUIDELINES

Why Set Up Standards/Rules/Guidelines: Clear standards ensure consistency and quality, providing a common framework for all team members to follow. E.g coding standards, testing procedure, SOPs, release and deployment proceedure

How to Set Up Standards/Rules/Guidelines: Develop these collaboratively with your team, document them clearly, and ensure they are accessible and regularly updated.

Note: Ensure rules are flexible enough to accommodate exceptions and encourage innovation within the framework.



5. CLEAR COMMUNICATION

Why Clear Communication: Effective communication prevents misunderstandings and aligns team efforts with project objectives.

How to Clear Communication: Utilize detailed project briefs, regular updates, and open channels for dialogue across all levels of the team.

Note: Remember that communication is a two-way street; be as open to listening as you are to sharing information.

6. EMPOWER YOUR DEVELOPERS

Why Empower Your Developers: Empowerment leads to greater ownership, innovation, and accountability among team members.

How to Empower Your Developers: Delegate meaningful tasks, provide the necessary resources for autonomy, and trust in their expertise to make decisions.

Note: Empowerment without support can lead to stress; ensure you provide guidance and a safety net for risks.

7. BUILD YOUR TEAM

Why Build Your Team: Developers like challenges so create an environment that they can grow and thrive in. Encourage and facilitate ongoing learning opportunities for your team. Promote activities that enhance team cohesion and morale.

How to Build Your Team: Invest in team-building activities, promote a culture of mutual respect, and celebrate team achievements.

Note: Team building should not force artificial relationships; respect the individuality and boundaries of team members.

7. HAVE ACCESS TO YOUR CODE/IP

Why Learn Some Tech: You must always have access to the code your team has developed. Access can be done via GITHUB or any other tool, but it helps in the unlikely event that the developer is unavailable to continue working.

How to Learn Some Tech: Use tools like GITHUB, et al to encourage sharing. This will mean you will always have access to all the knowledge products generated should you ever need to use another developer

Note: Code access does not mean micro-managing; trust your developers' expertise and use access for oversight, not constant interference.

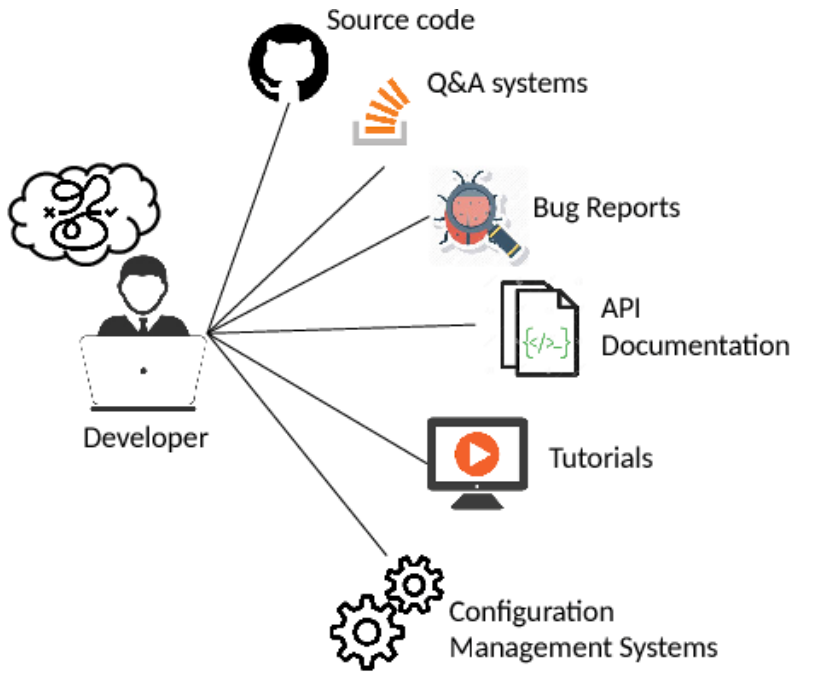


Fig. 1. Developers are overwhelmed by huge and

9. ELIMINATE SINGLE POINTS OF FAILURE

Why Eliminate Single Points of Failure: Reducing reliance on any one resources increases the resilience and reliability of your project.

How to Eliminate Single Points of Failure: Implement redundant systems, cross-train team members, and create comprehensive documentation.

Note: Avoid creating unnecessary complexity when building redundancies; simplicity can often be the key to reliability.

10. FIRE FAST

Why Fire Quick: Be quick to notice and recognize poor performance or a lack of your core values and behaviour. If they really do have the right attitude, then take the time to train the skill otherwise, let them go.

How to Fire Quick: Establish clear performance metrics, provide timely feedback, and if necessary, take decisive action to part ways.

Note: Firing should be a last resort, after support, coaching, and improvement efforts have been exhausted.

FEEDBACK?
THANK YOU