Stories

The Compiler: This story involves two groups of programmers, A and B.

- Group A developed a successful compiler for a language (L) on a specific computer (X).
- Group B was tasked with creating a compiler for a slightly modified version of the language (L+M) for a different computer (Y).
- Group B used Group A's compiler as a starting point and received comprehensive documentation and consultation from Group A.
- Despite this, Group B struggled to implement the extensions (M) effectively. Group A, with its deeper understanding of the compiler's underlying theory, could immediately identify flaws in Group B's proposed solutions. Group A then offered elegant solutions that worked within the existing compiler's framework.
- Years later, the compiler, further modified by other programmers without guidance from Group A, retained traces of the original structure. However, poorly-integrated additions had rendered it ineffective.
- **This story illustrates how documentation alone cannot convey the depth of understanding, the "theory," held by the original programmers. This theory is crucial for maintaining and adapting a program effectively over time.**

Real-Time System Installation and Maintenance

- This story centres on a large real-time system used for industrial monitoring. A dedicated team of programmers is responsible for installing, troubleshooting, and adapting the system for various clients.
- These programmers possess a deep understanding, a "theory," of the system's design and functionality, acquired through years of continuous engagement.
- When diagnosing faults, they rely primarily on their intuitive knowledge and the annotated program text, finding traditional documentation insufficient.
- Other programmer groups responsible for operating specific installations often face difficulties that stem from an incomplete grasp of the system's theory, even with access to documentation and guidance.
- **This example reinforces that a program's successful implementation, adaptation, and maintenance depend heavily on the programmers' deep, intuitive understanding, which transcends documentation.**

The theory leaves inside the programmer's head.

Ryle's Notion of Theory (knowing that vs knowing how)

Reciting a recipe vs being a chef.

- Entender el modelo.