Pattern	Brief Description
Rethinking Documentation	
MOST KNOWLEDGE IS ALREADY THERE	There is no need to record a piece knowledge when it is already recorded in the system itself.
PREFER INTERNAL DOCUMENTATION	The best place to store documentation is on the documented thing itself
FOCUS ON SPECIFIC KNOWLEDGE	Use documentation for the specific knowledge and learn the generic knowledge from trainings
ACCURACY MECHANISM	You can only trust documentation if there is a mechanism to guarantee its accuracy.
FUN ACTIVITY	To make an activity sustainable, make it fun
Knowledge Exploitation	
ACCURACY MECHANISM patterns	
SINGLE SOURCE PUBLISHING	Keep the knowledge in one single source of truth and publish from there when needed
RECONCILIATION MECHANISM	Whenever knowledge is repeated in more than one place setup a reconciliation mechanism to detect inconsistencies immediately.
CONSOLIDATION OF DISPERSED FACTS	Diverse facts put together become useful knowledge
TOOLS HISTORY	Your tools also record knowledge about your system.
READY-MADE DOCUMENTATION	Most of what you do is already documented in the literature.
Knowledge Augmentation	
AUGMENTED CODE	When the code doesn't tell the full story add the missing knowledge to make it complete.
DOCUMENTATION BY ANNOTATION	Extend your programming language using annotations for documentation purposes.
DOCUMENTATION BY CONVENTION	Rely on code conventions to document knowledge.
MODULE-WIDE KNOWLEDGE AUGMENTATION	Knowledge that spans a number of artifacts that have something in common is best factored out in one place.
INTRINSIC KNOWLEDGE AUGMENTATION	Only annotate elements with knowledge that is intrinsic to them
EMBEDDED LEARNING	Putting more knowledge into the code helps its maintainers learn while working on it
SIDECAR FILES	When putting annotation within the code is not possible put them next into a file to them.
METADATA DATABASE	When putting annotation within the code is not possible keep them in an outside database.
MACHINE ACCESSIBLE DOCUMENTATION	Documentation that is machine-accessible opens new opportunities for tools to help at design level
RECORD YOUR RATIONALE	Rationale behind decisions are some of the most important things to augment the code with.
ACKOWLEDGE YOUR INFLUENCES	The knowledge of the major influences of a team is a proxy of their mindset to better understand the resulting code.
COMMIT MESSAGES AS COMPREHENSIVE DOCUMENTATION	Carefully written commit messages make each line of code well-documented
Knowledge Curation	
DYNAMIC CURATION	It's not because all the works of art are already there in the collection that there is nothing to be done to make an exhibition out of it.
HIGHLIGHTED CORE (Eric Evans)	Some elements of the domain are more important than others
INSPIRING EXEMPLARS	The best documentation on how to write code is often just the code which is already there

GUIDED TOUR, SIGHTSEEING MAP (Simon Brown)	It is easier to quickly discover the best of a new place with a guided tour or a sightseeing map.
Automating Documentation	
LIVING DOCUMENT	A document that is evolving at the same pace as the system it describes
LIVING GLOSSARY	The code as the reference of the domain language of the system
LIVING DIAGRAM	A diagram that you can generate again on any change so that it's always up-to-date.
ONE DIAGRAM	ONE STORY One diagram should only tell one specific message.
Runtime Documentation	
VISIBLE TEST	Tests that produce a visual output for human review in a domain-specific notation
VISIBLE WORKINGS (Brian Marick)	Working Software as Its Own Documentation
INTROSPECTABLE WORKINGS	Your code in memory as a source of knowledge.
Refactorable Documentation	
CODE AS DOCUMENTATION	Most of the time the code is its own documentation.
INTEGRATED DOCUMENTATION	Your Integrated Development Environment (IDE) already fulfils many documentation needs.
PLAIN-TEXT DIAGRAMS	Diagrams that cannot be genuine living diagrams should be created from plain-text documents to ease their maintenance.
Stable Documentation	
EVERGREEN CONTENT	Evergreen Content is content that remains useful without change for a long time
PERENIAL NAMING	Some names last longer than others.
LINKED KNOWLEDGE	Knowledge is more valuable when it is connected provided the connections are stable.
LINK REGISTRY	An indirection that you can change to fix broken links in one single place.
BOOKMARKED SEARCH	Search made into a link is more stable than a direct link
BROKEN LINK CHECKER	Detecting broken links as soon as possible helps keep it trusted.
INVEST IN STABLE KNOWLEDGE	Stable knowledge is an investment that pays back over a longer period of time.
How to Avoid Documentation	
CONVERSATIONS OVER FORMAL DOCUMENTATION	
WORKING COLLECTIVELY AS CONTINUOUS KNOWLEDGE SHARING	Working Collectively is an opportunity for Continuous Knowledge Sharing.
COFFEE MACHINE COMMUNICATION	Not all exchange of knowledge has to be planned and managed. Spontaneous discussions in a relaxed environment often work better and must be encouraged.
IDEAS SEDIMENTATION	It takes some time to find out whether a piece of knowledge was important or not.
THROW-AWAY DOCUMENTATION	Documentation that's only useful for a limited period of time before it can be deleted.
ON-DEMAND DOCUMENTATION	Document what you've seen necessary to be documented.
ASTONISHMENT REPORT	Newcomers' superpower is to be bring a fresh perspective.
INTERACTIVE DOCUMENTATION	Documentation that tries to emulate the interactivity of a conversation.

DECLARATIVE AUTOMATION	Every time you automate a software task, you should take the opportunity to make it a form of documentation as well.
ENFORCED GUIDELINES	The best documentation does not even have to be read if it can alert you at the right time with the right piece of knowledge
CONSTRAINED BEHAVIOR	Don't document influence or constraint the behaviour instead.
REPLACEABILITY-FIRST	Designing for replaceability reduces the need to know how things work.
CONSISTENCY-FIRST	Being consistent reduces the need for documentation.
Beyond Documentation: Living Design	
LISTEN TO THE DOCUMENTATION	Documentation as a signal to spot opportunities for improvements.
SHAMEFUL DOCUMENTATION	The presence of a free comment is often a signal of a shameful behavior in the code.
DELIBERATE DECISION-MAKING	The path to better design and better documentation starts by taking more decisions deliberately.
HYGIENIC TRANSPARENCY	Transparency leads to higher hygiene because the dirt cannot hide.
WORD CLOUD	A word cloud of the identifiers in the code should reveal what the code is about
SIGNATURE SURVEY (Ward Cunningham)	Looking at the code at some level of details to reveal its shape
DOCUMENTATION-DRIVEN	Start by explaining your goal or end result e.g. how your system will be used
ABUSING LIVING DOCUMENTATION (ANTI-PATTERN)	Don't go dogmatic about Living Documentation and keep subordinate to delivering value for your users.
LIVING DOCUMENTATION PROCRASTINATION	Have fun in your Living Documentation tools to avoid having too much fun in your production code.
BIOGRADABLE DOCUMENTATION	The goal of a documentation should be to make itself redundant.
DESIGN SKILLS EVERYWHERE	Learn and practice good design; it's equally good for your code and for your documentation for the same reasons.
Living Architecture	
DOCUMENT THE PROBLEM	Every time you document a solution without explaining the problem it attempts to solve, God kills a kitten.
STAKE-DRIVEN ARCHITECTURE	Is your biggest challenge on the domain understanding on a quality attribute or on socio-technical aspects?
EXPLICIT QUALITY ATTRIBUTES	Friends don't let friends guess the quality attributes the system was designed for.
ARCHITECTURE LANDSCAPE	Organizing the multiple documentation mechanisms into a consistent whole for easier navigation.
DECISION LOG	Keep the major system-wide decisions into a decision log
FRACTAL ARCHITECTURE DOCUMENTATION	Your system is made of systems. Organize your documentation accordingly
ARCHITECTURE CODEX	Documenting the way you take decisions enables decentralized decision-making
TRANSPARENT ARCHITECTURE	Architecture is for everyone, provided they have access to the information.
ARCHITECTURAL REALITY CHECK	Making sure that the implementation of the architecture matches its intent.
TEST-DRIVEN ARCHITECTURE	The ultimate living architecture is test-driven
SMALL-SCALE SIMULATION AS DOCUMENTATION	Documenting a large system with a smaller version of itself.
SYSTEM METAPHOR (Kent Beck)	A story that everyone, customers, programmers and managers, can tell about how the system works.

Introducing Living Documentation	
UNDERCOVER EXPERIMENTS	Start with safe-to-fail experiments without much publicity.
MARGINAL DOCUMENTATION	New practices can usually only be applied to new work.
COMPLIANCE IN SPIRIT	A Living Documentation approach can comply to even the most demanding compliance requirements by aiming for the spirit instead of aiming for the letter.
Documenting Legacy Applications	
FOSSILIZED KNOWLEDGE	Legacy systems should be considered blindly as a reliable documentation.
BUBBLE CONTEXT (Eric Evans)	Even in a legacy system you want to work as much as possible in the ideal land where everything is nice and clean.
SUPERIMPOSED STRUCTURE	Relate the desirable structure to the existing, less desirable one
HIGHLIGHTED STRUCTURE	Making a superimposed structure visible in relation with the existing source code
EXTERNAL ANNOTATIONS	Sometimes we don't want to touch a fragile system just to add some knowledge in it
BIODEGRADABLE TRANSFORMATION	Documentation of a temporary process should disappear with it when it is done
MAXIMS (Serge Demeyer)	Big changes to legacy systems are made by a number of people who share common objectives; use maxims to share the vision.
ENFORCED LEGACY RULE	Legacy transformations can last longer than the people doing them; automate the enforcement of the big decisions to protect them