* **Modern Code Review: A Case Study at Google** <https://moodle.gla.ac.uk/pluginfile.php/2476949/mod_resource/content/1/Sadowski2018ModernCodeReview.pdf> - about code review practices at google – survey of current practices and how satisfied the devs are with them. Devs supper happy about reviewing very small frequent changes with quick turnaround time. Code reviews good, small commits good.
* **Design and code inspections to reduce errors in program development** <https://moodle.gla.ac.uk/pluginfile.php/2476975/mod_resource/content/1/fagan76design.pdf> how to do code reviews, how to spot errors. Inspections increase productivity and final program quality. Inspections improve project management and process control
* **Inspections—Some Surprising Findings** <https://moodle.gla.ac.uk/pluginfile.php/2476981/mod_resource/content/1/glass99inspections.pdf> code inspections extremely effective number of participants in code review should be 2 or 3. Inspection meetings not useful meetings should only be used to discuss findings
* **A metrics suite for object oriented design** <https://moodle.gla.ac.uk/pluginfile.php/2498239/mod_resource/content/1/chidamber94metrics.pdf> OO metrics help non technical managers understand the implementation, metrics can help find the best OO design for organisation’s goals OO may hold solutions to the software crisis.
* **Introducing BDD** [**https://dannorth.net/introducing-bdd/**](https://dannorth.net/introducing-bdd/) **how to create good software tests**
* Maintaining Behaviour Driven Development Specifications: Challenges and Opportunities <https://moodle.gla.ac.uk/pluginfile.php/2528660/mod_resource/content/2/Binamungu2018MaintainingBDDChallengesOpportunities.pdf> BDD used by many software teams. Creates living documentation. Management of documentation of time is hard. allow the requirements to be executed to check whether the production code implements the requirements successfully or not.
* **An Empirical Evaluation of Mutation Testing for Improving the Test Quality of Safety-Critical Software** <https://moodle.gla.ac.uk/pluginfile.php/2542889/mod_resource/content/1/06298894.pdf> Mutation testing in this study has identified shortfalls in test cases which are too obscure to be detected by manual review. Mutation testing also offers a consistent measure of test quality which peer review cannot demon-strate. Analyzing test results and determining equivalent mutant behavior is still a manual overhead and therefore requires further investigation.
* **Hints on tata test selection:** <https://moodle.gla.ac.uk/pluginfile.php/2542888/mod_resource/content/1/demillo78hints.pdf> another one on testing. Not relevant
* **Software release build process and components in ATLAS offline** <https://moodle.gla.ac.uk/pluginfile.php/2546913/mod_resource/content/1/obreshkov10software.pdf> Performing the builds and tests every night accelerates software development and increases its quality. The combination of nightly and numbered releases helps the support of development and production activities. Our ability to rapidly fix bugs and provide patches for problems has proven to be quite efficient for the validation and production efforts. However that requires dedicated hardware and manpower resources to keep the infrastructure up and running

# **What's Wrong with Software Reuse?** <https://www.stickyminds.com/article/whats-wrong-software-reuse> the theory-based prediction that software reuse will lead to unbounded improvements in both development productivity and software quality. "It ain't so," he says here. But he certainly wishes that it were.

* **Flight 501 Failure** [**http://www.di.unito.it/~damiani/ariane5rep.html**](http://www.di.unito.it/~damiani/ariane5rep.html)The extensive reviews and tests carried out during the Ariane 5 Development Programme did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure.
* <http://www.vincehuston.org/dp/> list of very many design patterns and their uses

# Catalog of Patterns of Enterprise Application Architecture <https://martinfowler.com/eaaCatalog/> list of patterns and what they’re good for

* <https://hillside.net/patterns/patterns-catalog> list of patterns and what they’re good for
* **Planning the software industrial revolution** <https://moodle.gla.ac.uk/pluginfile.php/2554249/mod_resource/content/1/00060587.pdf> benefits of OO, test procedures
* component Primer <https://moodle.gla.ac.uk/pluginfile.php/2554247/mod_resource/content/1/p27-hopkins.pdf> Component-based software development represents an important stage in the maturation of the field of software engineering. It shifts the focus from new software development to the integration of existing components to perform new tasks. At the same time it addresses the issues of large-scale system develop-ment in the areas of coupling, distribution, and multiple platforms. As the component marketplace forms, it becomes one of so-called “increasing returns” whereby the components become more valuable as more become available, precisely because they start to realize the potential of the component-based ideal
* Refactoring <https://www.refactoring.com/> good things about benefits of refactoring
* Code [http://codev2.cc/download+remix/Lessig-Codev2.pdf intellectual property / privacy / limits of open code/ regulating code (400](http://codev2.cc/download+remix/Lessig-Codev2.pdf%20intellectual%20property%20/%20privacy%20/%20limits%20of%20open%20code/%20regulating%20code%20(400) pages)
* THE FUTURE OF IDEAS <http://www.the-future-of-ideas.com/download/lessig_FOI.pdf> on software copyright and related issues (400 pages) the argument of this book is that always and everywhere, free resources have been crucial to innovation and creativity; that without them, cre-ativity is crippled. Thus, and especially in the digital age, the central ques-tion becomes not whether government or the market should control a resource, but whether a resource should be controlled at all. Just because control is possible, it doesn’t follow that it is justified. Instead, in a free soci-ety, the burden of justification should fall on him who would defend sys-tems of control.
* Report of the Inquiry Into The London Ambulance Servic <https://moodle.gla.ac.uk/pluginfile.php/621363/mod_resource/content/4/swtrha93report.pdf> software wasn’t complete when it was released, staff weren’t trained, there was no paper backup. 62 pages of findings on poorly finished software, might be useful
* Experimenting with Realism in Software Engineering Team Projects: An Experience Report. <https://moodle.gla.ac.uk/pluginfile.php/2302985/mod_resource/content/1/simpson2017experimenting.pdf> very useful article on the justification about why team project 3 is run with real customers and the rationale behind the way it’s done. 23 further references that could be very useful.
* SCRUM Development Proces <https://moodle.gla.ac.uk/pluginfile.php/2417391/mod_resource/content/2/schwaber1995scrum.pdf> CRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle

# 5 Useful Tips For A Better Commit Message <https://thoughtbot.com/blog/5-useful-tips-for-a-better-commit-message> why do good commit messages

## Twenty dirty tricks to train software engineers <https://repository.lboro.ac.uk/articles/Twenty_dirty_tricks_to_train_software_engineers/9405065> limitations of academic groupwork – relevant for working with a real client

* <http://www.mountaingoatsoftware.com/agile/user-stories> about doing user stories
* Planning Poker or How to avoid analysis paralysis while release planning <http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf> how to evaluate how long a task will take quickly