Software development is rife with acronyms, abbreviations and jargon. Those in technical professions often employ acronyms in particular to add validity to their own theories, sound important in meetings and obfuscate their own incomplete knowledge of complex topics. This leads to painful constructs such as recursive acronyms (e.g., PIP, for “PIP Installs Packages”) and backronyms, retronyms, et cetera.

Although you should be suspicious of anyone using jargon in a meeting (particularly if he/she pronounces it loudly and then surveys the room for a reaction), acronyms are often used effectively as convenient shortcuts to complex theories and principles. Clients often use these words conversationally, so be sure to familiarize yourself with some of the more commonly used specimens.

What follows is a primer for some high level conceptual acronyms you might hear. Once you understand them, inject them yourself in conversations - used effectively, they may bring a group to a common understanding very quickly. As you acquire an ever-larger repertoire of jargon, you will find it easier to speak in acronyms than complete sentences, and one day someone will tap you on the shoulder and tell you that you are an architect!

* DRY - Don’t Repeat Yourself. This may be the most important acronym in software engineering, because it guides your behavior in almost every stage of the software lifecycle. It directs us to avoid copy-paste code; deploy software builds automatically instead of by hand; aggressively address repeating bad behaviors identified by the team during sprint retrospectives; for managers, encourages (some of) us to use linked cells in our Excel spreadsheets. DRY gives developers the license to do the right thing by being lazy. It doesn’t get any better than that.
* YAGNI - You Ain’t Gonna Need It. This principle directs software developers not to introduce complexity into our architectures. Take, for example, admonition, “Premature optimization is the root of all evil.” YAGNI states that you should plan for performance issues but not optimize for performance until you are sure that you need to. For project managers, YAGNI is a mantra that can be repeated to clients and project teams that try to add superfluous features that are out of scope. YAGNI is related to the KISS principle (Keep It Simple, etc.)
* RDD - Resume Driven Design. I know, this isn’t an acronym, and it’s not in wide usage, but it accurately reflects the tendency of some developers to architect with new, often immature technologies because they think it will look good on their resumes. I consider this to be an anti-pattern.
* ACID - Atomicity, Consistency, Isolation, Durability. These are the qualities expected in a transactional system, especially relational databases. In a nutshell, it means that a transaction is all-or-nothing, ensures data is valid in multi-user scenarios, does not affect other users, and should save its state. An alternative to ACID is described by the highly contrived acronym BASE - Basically Available, Soft State, Eventually Consistent. BASE describes the transaction characteristics of many NoSQL databases, which eliminate much of the locking that provides consistency but slows down database operations, allows data to persist asynchronously (“eventually”) and is may not always present the latest version of data.
* CAP (Theorem) - Consistency, Availability, Partition tolerance. These are properties of databases, from which you can pick two at the expense of the third. For example, MongoDB offers availability and partition tolerance at the expense of consistency. CAP becomes important when considering which databases to incorporate in your architectures.
* SOLID - Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion. All qualities followed in application design that allow for an extensible, maintainable system. If you want to improve your standing on your project team, find an opportunity to accuse a teammate of violating the Liskov Substitution Principle during a code review.
* COTS - Commercial Off-The-Shelf. This is software you buy and configure, rather than code from scratch. Its counterpart is FOSS (Free Open Source Software).
* SOX - Sarbanes Oxley. SOX legislation compels publicly traded financial institutions to implement strong data security, ensure integrity of data and audit any system events that could compromise access to data. You will need to consider SOX requirements when designing your architecture.
* DRY - Don’t Repeat Yourself.

This is just a small sample of acronyms for your growing arsenal. Remember, the only thing better than using jargon yourself is asking someone else to explain their jargon and watching them squirm.