Kristof Kovacs
Software architect, consultant

A forgotten military standard that saves weeks of work (by providing free project management templates)

Nobody loves to write documentation, but with age and experience, IT project managers usually come to accept that it can be useful sometimes, even necessary. In healthy doses.

Now, if you're doing small to medium software projects, the US military might not be your usual place to turn to for inspiration. But in one case, back in '94, they did create something truly wonder-, power-, and useful. (Then they've thrown it away and replaced it with a complex and much less useful hybrid IEEE and ISO monster. But there's an interesting twist -- more on that later.)

This gem is inspirationally called **"MIL-STD-498"**.

It was created to fix the problems with an older standard (the DOD 2167A, if you need to know), for example its insistence on using the waterfall model, or requiring huge demonstration events that stopped development for weeks. (People unofficially called those *"dog & pony shows"*.)

MIL-STD-498 can help you if you are a developer dealing with clients, but also if you're a project manager dealing with suppliers. I personally had used it in both situations.

By the end of this article, you'll be proficient enough in it to brag about it. And to know how to use it for various purposes when the need arises the next time.

So, why is MIL-STD-498 so cool and unique?

1. It's in the public domain, so you're free to use it. Unlike IEEE or ISO standards, which you'd have to buy.

Like the Internet RFCs, it's out there. Actually, it's right here (zip download) (https://github.com/kkovacs/MIL-STD-498-templates-html/archive/master.zip). I had an assistant convert the templates (called "DIDs", Data Item Descriptions) to friendly clean HTML (with nested header levels and without messy formatting), so you can easily use it with any word processor. The original Word-97 files even MS Office doesn't seem to read correctly anymore.

2. It's completely self-sufficient. It invokes no other standard. It's stand-alone.

Some standards get entangled with others by invoking various other standards, requiring you to include, import (and often buy) documents that belong to those other standards, not only resulting in a chaotic mess, but also making it hard to initially estimate how big the work of documentation will be.

3. It is essentially a collection of document templates. And not only a table of contents (although that already would be a great free project management template), but also "help" paragraphs explaining what to include, how to approach the topic, what to reference, etc.

Even if you're not doing full-blown documentation, these project management templates ("DIDs") are useful as checklists.

4. It's compatible with the latest. You might be thinking, *"ok, but still, an outdated standard? Why should I even care?"* But, you have to know that the MIL-STD-498 later got "civilized" as J-STD-016, which became part of the current IEEE/EIA 12207 (which also includes, but is not the same as ISO 12207, just to demonstrate how chaotic it can get in the world of standards). So, basically, if you work with MIL-STD-498, you're creating documents that perfectly fit into the modern 12207! (Which, by the way, doesn't have document templates for you, even if you buy it. Just saying.)

5. It's quite flexible. The creators of the standard have understood that projects come in various shapes, sizes, and colors. So, the MIL-STD-498 can officially be customized to the project. They call it "tailoring", and they've even written a full guide on how to do it officially. But in most cases, if you're not a DoD supplier, just feel free to leave out the parts you feel are not important for your project (for example, the part about how the item is supposed to work when there are enemy explosions nearby.)

Also, it allows the use of electronic tools instead of written word, when appropriate. Its thinking is that the point is that you have the stuff, not that it must be written in a word processor.

6. It supports multiple program strategies. It clearly says that there are three ways of doing projects: "Grand design" (also known as, waterfall), "Incremental" (which most Agile projects do), and even "Evolutionary" (which includes exploratory projects like prototypes).

So, how does one actually use it?

Well, there's the 344 pages long *"Application and Reference Guide"* (Part1 (http://www.abelia.com/498pdf/498GBAR.PDF), Part2 (http://www.abelia.com/498pdf/498GBAR5.PDF)), and the 99-page strong "Overview and Tailoring Guide (http://www.abelia.com/498pdf/498GBOT.PDF)". Or, you can start using the templates (the "DIDs") right away just after reading this short introduction from me.

(I'm listing the documents in the order they usually get written. Of course, certain projects might be different. Or you just need one template to document one aspect -- feel free to.)

Request For Quotation phase

- ***OCD - Operational Concept Description (/stuff/MIL-STD-498-templates-html/OCD.html)*** . This describes in the client's language what the project is about, what is wrong with the current situation, how will the system improve it. If you're a developer, it's best used as a checklist for questioning the client about the project details; if you're a project manager, use it as a template to write a very good description of your planned project for the suppliers.

(Pro tip: As a purchaser, you get lower prices if your description is well thought out, and not vague. The developers' biggest risk factor is actually *you*: They know how they can program, but they don't know how difficult you are. A good, concise project description shows that you have already had put ample thought into the project, which is a good sign, and they don't have to pad the estimates just in case you turn out to be the *Client From Hell Who Orders A Bicycle But Actually Needs A Nuclear Submarine*.)

- ***SDP - Software Development Plan (/stuff/MIL-STD-498-templates-html/SDP.html)*** . It just describes the very basics of how the development will happen: languages, tools, source control, trouble ticketing, change management, testing, validation, phases, timetable, etc. It's best used as a checklist of things that should go into a contract.

These two are usually enough for an initial ballpark price quoting phase.

There's one more that *has* to be done before contracting - just so it's clear for everyone what is the acceptance criteria:

- ***STP - Software Test Plan (/stuff/MIL-STD-498-templates-html/STP.html)*** . It describes in general how and where the testing (especially the acceptance testing) will happen, without going into details on individual tests. (Those come later in the unfortunately named STD document.)

Now when the project is about to start, proceed to the next ones:

System design phase

- ***SSS - System/Subsystem Specification (/stuff/MIL-STD-498-templates-html/SSS.html)*** and ***SRS - Software Requirement Specification (/stuff/MIL-STD-498-templates-html/SRS.html)*** . There can be only one SSS, but if needed, there can be an SRS for each Subsystem. (I usually only do this sub-document thing if the different subsystems are the responsibilities of different suppliers.) The SSS and SRS are basically the same and differ only in minor wording -- in theory, the SSS might contain hardware descriptions too, while the SRS can only describe software.

The SSS (and maybe SRSes) is how one describes the architecture of the system. This is best done in cooperation (in "Joint Application Design", or JAD sessions, if you want a military-sounding buzzword) between the client and the developer. Use it as a template.

- ***IRS - Interface Requirement Specification (/stuff/MIL-STD-498-templates-html/IRS.html)*** . Only needed if the system will connect to one or more external systems. Best used as a checklist of what to collect from (usually) third parties. (One can rarely dictate the format of third party documentation.)

The SSS (or SRS) and the IRS should contain enough information that the software development can actually be done. It should start now. The following documents are to be done *during* the development:

Development phase

- ***SSDD - System/Subsystem Design Description (/stuff/MIL-STD-498-templates-html/SSDD.html)*** (or ***SDD - Software Design Description) (/stuff/MIL-STD-498-templates-html/SDD.html)*** and ***IDD - Interface Design Description (/stuff/MIL-STD-498-templates-html/IDD.html)*** . These describe the evolving, and in the end the final design of the system and the subsystems and the interfaces. These are basically counterparts with the Specifications:

  SSS - SSDD
  SRS - SDD
  IRS - IDD

The templates that end in "S" (as in Specification) are supposedly written (or at least, the information is given) by the customer; while the ones ending in "DD" (as in "Design Description"), are written by the developer.

This might sound nitpicking at first, but actually, it's very important, because MIL-STD-498 does away with the usual thinking that the customer says "what" to do, and the developer decides "how" to do things. It says that if it's in the Specifications then the developer must comply with it, doesn't matter if it's a "what" or a "how"; on the other hand, if something is *not* in the specification (be that a "what" or "how"), then it's up to the developer, but should be documented in the "DD"-s. See how beautiful the separation of responsibilities is?

This separation fits both knowledgeable and less experienced customers, since they can specify as much or as less as they wish, or can.

- ***DBDD - Database Design Description (/stuff/MIL-STD-498-templates-html/DBDD.html)*** . Just what it sounds. Actually, this one is often better done with a "create table, create index, etc" SQL script or with one of the fancy database design tools than in a text editor. (That's what MIL-STD-498's spirit says: Don't you *dare* to copy the SQL into the documentation just to make it look thick. Just give me the file.)

- ***STD - Software Test Description (/stuff/MIL-STD-498-templates-html/STD.html)*** . This well-named template describes the information missing from the STP (the Test Plan) -- namely, the individual tests. Best used as a checklist, the tests themselves can usually be best described in a spreadsheet format.

Tests should cover every important functionality of the system, and also don't forget that tests are needed for two purposes: 1) to check functionality, and 2) to test against possible attacks.

In my experience, tests are best written both by the customer and the developer separately, and then merged together. (There will be overlaps that need to be reconciled, but there will also be interesting differences in approach.) Also, I find it best if test descriptions are written once at the beginning of the development, then retouched and expanded again as the end is nearing (and in between if the development is long enough or is separated to phases).

Test descriptions are a very good way to find hidden requirements. Don't miss the opportunity, early in the process.

Testing phase

- ***STR - Software Test Report (/stuff/MIL-STD-498-templates-html/STR.html)*** . This is basically the output of doing the STDs according to the STP. Best read once for scraping ideas, and actually done by adding check marks and notes to a copy of the STD spreadsheet.

Manuals

- ***SUM - Software user manual (/stuff/MIL-STD-498-templates-html/SUM.html)*** . Very good template. Use it. (And don't forget, it's best to document by user functions, not by features. But this is just my advice, it's not in the standard.) Can be omitted if there is no user interface.
- ***SIOM - Software Input/Output Manual (/stuff/MIL-STD-498-templates-html/SIOM.html)*** . It's best used as a template for documenting APIs. Omit if there is no API.
- ***SCOM - Software Center Operator Manual (/stuff/MIL-STD-498-templates-html/SCOM.html)*** . A template to document the operation of the system. It's to be done if someone else will do the hosting.

Delivery phase

- ***SPS - Software Product Specification (/stuff/MIL-STD-498-templates-html/SPS.html)*** . Best used as a checklist of things expected to be delivered.

- ***SVD - Software Version Description (/stuff/MIL-STD-498-templates-html/SVD.html)*** . This is basically a "change log" template. Use when delivering changes to a system already in operation.

Rarely used DIDs

Templates that are rarely used -- at least in my line of work:

- ***STrP - Software Transition Plan (/stuff/MIL-STD-498-templates-html/STRP.html)*** . It describes how to transfer the support functions to the customer's internal support agency. It would basically be a knowledge base for user support personnel.
- ***COM - Computer Operation Manual (/stuff/MIL-STD-498-templates-html/COM.html)*** . Well, unless you're developing a whole new computer. In that case, you should totally use this one.
- ***CPM - Computer Programming Manual (/stuff/MIL-STD-498-templates-html/CPM.html)*** . I guess it's a useful starting point if your software has it's own programming language.
- ***FSM - Firmware Support Manual (/stuff/MIL-STD-498-templates-html/FSM.html)*** . I suppose it's useful for embedded systems. Which I haven't done lately.

Conclusion

If you are a military supplier, or intent on doing this the "perfect" way, then definitely read the two big manuals. And the Appendices. And tailor the documents by the book.

But just by having the above information in your head (or in your browser), and having access to these templates, I'll say that you're already 85% of the way. Most "normal" project managers, even with relatively big projects, will never need to know more about MIL-STD-498 than this. Insert it into your own process.

I wish that you use this little gem with success - and start to hate doing documentation just a little less. :)

Shameless plug: I'm a freelance software architect (resume) (/resume) , have a look at my services (/services) !