How to Modernize the Procurement Process for Major IT Projects
**Human-centered design.** All work needs to be centered around satisfying the needs of end-users, who test it at every step of the way — work is only performed if actual end-users approve it, or in the service of their identified needs, and that work isn’t finished until the users say it is. Do what the actual relevant humans need, and not what their boss’s boss thinks they need.
**Agile software development.** Plan a software projects only in the broadest of strokes, with only a brief description of the goals. Work is done in 2-week cycles. Don't look ahead more than 2 weeks, because you have no idea what the future holds. Functioning software is delivered every 2 weeks, no excuses — fully tested and documented, ready to be used. If the vendor is bad, you fire them and hire a new one, who picks up where the old one left off.
Think of planning a drive around the country with your family — you could try to schedule all 6 weeks in advance, plotting out your exact route, reserving hotel rooms, buying tickets for attractions, etc., but you know that will never work, because reality is too messy — instead, you’d plan for the highlights you want to hit, put together a broad route, and figure out specifics as you go.
**DevOps.** This is the practice of automating the work that goes into testing and deploying software, merging software development and system operations. All testing and deployment is automatic. Developers are responsible for their code running properly on servers, desktops, etc. — they can’t throw it over the fence when they’re done and say “hey, it works for us.”
Build with loosely-coupled parts. Large software projects are doomed to fail — they will collapse under the weight of administration — so they must be broken down into a handful of small, quasi-independent software projects. Each modular component communicates with others via simple standards. Parts can always be swapped out. Vendors don’t need to coordinate. This is how power outlets work.
This is how Legos work.
This is how cloud works. All cloud-based development is using loosely-coupled parts.
Software talks to other software via an application programming interface (API). Here’s an actual “conversation” with an API — this is how one email server talks to another to send a message. You can have one vendor build an email server, and the other build an email client, and those vendors never need to talk to each other, because they’re using this common API.
**Modular contracting.** by employing HCD, Agile, DevOps, and loosely-coupled parts, now you can break up one huge, risky contract into a handful of small, cheap contracts. You want the contract to be small enough that you'll have no compunction about firing a non-performing vendor, knowing that the rest of the vendors will continue working, so your total loss of velocity is minimal. By using small contracts, each contract can come in within your state’s simplified procurement threshold, meaning that agencies can write an RFP, publish it, review responses, and award a contract all within 60 days. Rule of thumb: an agile development team (5–9 people) costs $1–2M/year.
These Things Are Bad

1. Detailed requirements document about the exact functionality that will be required.
2. A vendor spending months or years developing software, only delivering value when the project is done.
4. Contracts for more than a couple of million dollars, or with a PoP longer than a year.
5. Spending more than $10 million in total.

These are the five things you want to make sure that agencies don’t do.
The great majority of government software projects either fail outright or fail to achieve their goals. Only 13% are successful.
Small government software projects are much more likely to succeed. This survey of small government software projects (less than $6M) find that the majority are successful.

The resolution of small government software projects from fiscal 2010 to 2014 within the CHAOS database. Small is defined as labor cost less than one million euros or dollars. Classic CHAOS metrics define successful projects as on time, on budget, and are on target. Challenged projects are over budget, late, and/or have an unsatisfactory target. Failed projects are projects that were either canceled prior to completion or not used after implementation.
These Things Are Good

1. Rethink risk — beginning with no more big contracts.
2. Consider switching costs up front.
3. Plan to share your software with other states, and communities within your state.
4. Expand your vendor pool.
5. Hire tech talent and have them at the table from the start.
6. Leadership should lead.

These are the six rules to keep in mind to make your software projects a success.
Robin Carnahan
robin.carnahan@gsa.gov

Waldo Jaquith
waldo.jaquith@gsa.gov