

Up and Running Software – The Development Process

Success Determination, Adaptive Processes, and a Baseline Approach

About This Document:

Thank you for requesting more information about Up and Running Software’s development workflows. We would like to introduce our processes with the following document, as well as present how a software development process designed for your needs can be arrived at.

The visualization below presents how a software project typically evolves in our experience. This document focuses in on the Plan, Work, and Communication components of the software project (rows 2 and 3), which exist to ensure that a system that people like and use is created.

The Simplistic Software Project from Inception to Use



The approach we take in presenting this material helps uncover what you consider important in a process and a project, sets the expectation that processes within a project will almost certainly evolve and adapt as the context changes and your needs change, and then presents our baseline approach, as well as some other important project considerations.

If you have feedback or questions about anything as you go through this document, we would be pleased to hear from you at any time.

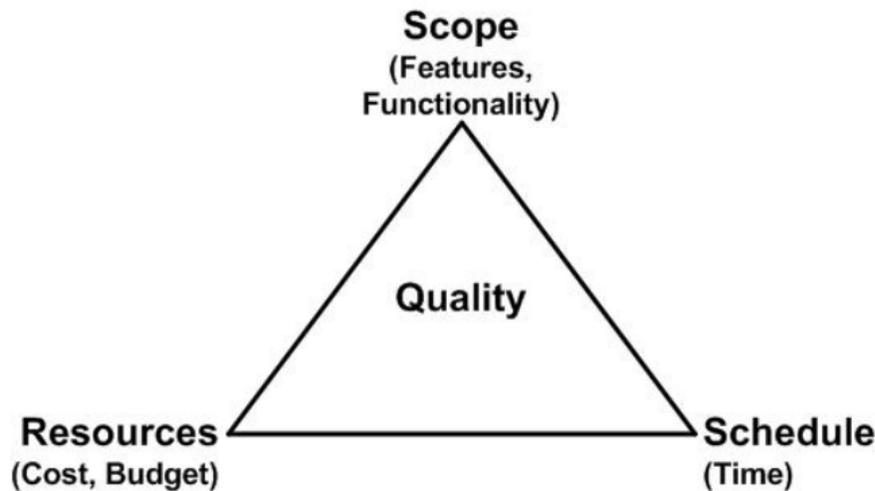
Process Goals – What Determines Success?:

The primary goal of any process is results, how **YOU** determine success. It's important to define explicitly what success is to you. If you and we don't do that, then the process will likely not achieve this from the start, though it'll likely get there over time with adjustments. Here are the common success criteria for a software development project, which should be thought through and prioritized:

- Architecture: how well the system is designed for future growth and extensibility.
- Build vs. Buy/Open Source: how much of the code will be purchased or community-derived versus built from scratch.
- Customer involvement: whether you want to be actively involved, or be involved as little as possible.
- Money (Resources, Cost, Budget): how much money is spent to get the work done.
- Quality: where within the spectrum of proactive-to-reactive quality assurance you wish to be.
- Scope: how much or how little work will be done. This could be described as “quantity of output” and “depth or quality of output”.
- Speed (Schedule, Time): how fast the work is done.
- Standardization: to what degree the process and/or code should be standardized per your company's needs and preferences.
- ...many more.

Scott Ambler, a leader in the agile arena, has presented this concept of competing goals nicely by focusing on perhaps the three most important goals of a project: scope, resources, and schedule. In one section, he writes, “Recognize that the iron triangle must be respected. The iron triangle refers to the concept that of the three critical factors – scope, cost, and time – at least one must vary otherwise the quality of the work suffers.

Nobody wants a poor quality system, otherwise why build it? Therefore the implication is that at least one of the three vertexes must be allowed to vary. The problem is that when you try to define the exact level of quality, the exact cost, the exact schedule, and the exact scope to be delivered you virtually guarantee failure because there is no room for a project team to maneuver.”



Copyright 2003-2006 Scott W. Ambler

Source for quote and image: <http://www.ambysoft.com/essays/brokenTriangle.html>

How these are prioritized and assigned relative weights by you is a subjective determination. There is no right or wrong approach; it's just a matter of what works better for you. However, it's important to recognize the competing relationships between these success criteria, for example:

- If you want the project to be done quickly, there are several ways to do that: 1) Increase the number of team members, 2) decrease the scope, 3) loosen the quality restrictions, 4) use something that exists already (buy or use open source software), and more. Of course, all of these affect money, quality, scalability, and more.
- If you want the project to be done cheaply, you'll need to explore if you want a lot done in an "ok" manner or a smaller amount done really well. What will influence your decisions are quality, scalability, timeline, and more.
- If you want the system to work seamlessly and be architecturally sound, you'll need to invest in the design of the architecture, carefully consider whether the work should be built or bought/found within the open source community, and plan in senior-level development reviews of all the code throughout the process. All of this takes time, which increases the timeline and costs in the short-term (in the long-run, the total cost of ownership (TCO) will be low for a well-designed system, whereas the TCO of a poorly-designed system may be less for the first year or two, but will cost more, usually multiples more, over the system's total life).

Once the success criteria are known, then your preferred process can be defined. Some examples:

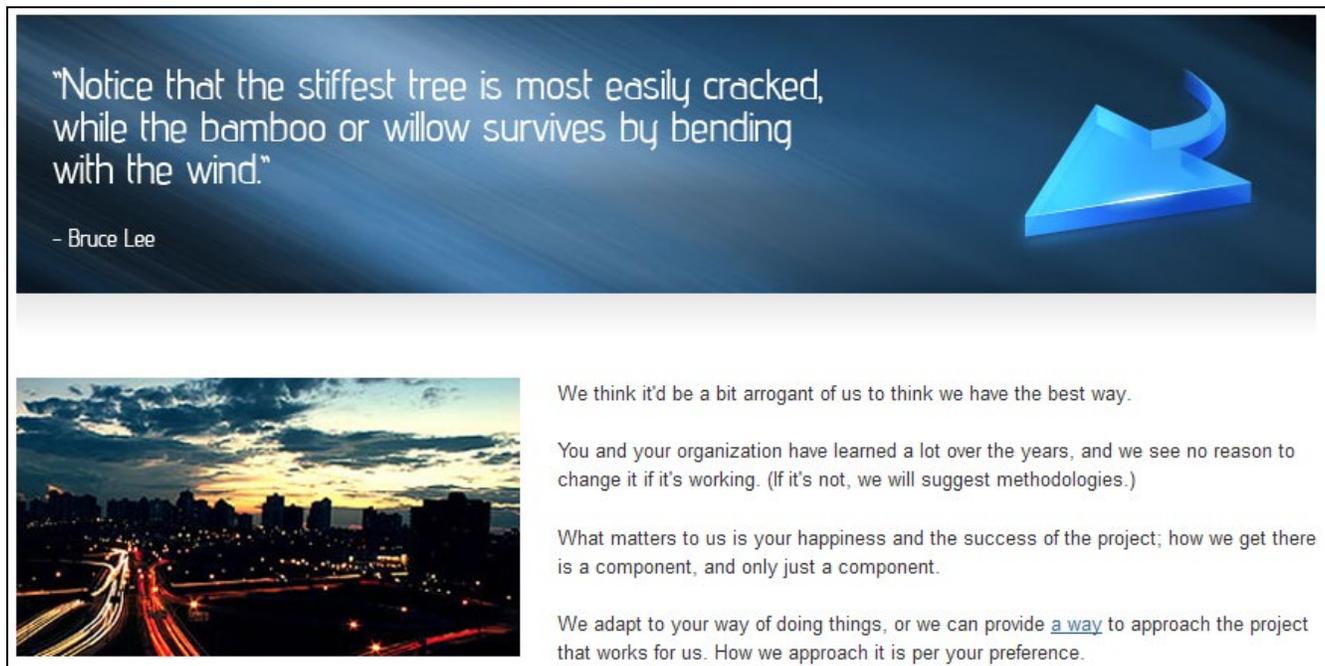
- Speed and quality should take precedence: processes will be defined to move as fast as possible, and ensure the greatest level of quality is adhered to. This will increase costs.

- Cost minimization should take precedence: processes will be defined to control costs, which will slow down the speed of development. This means fewer resources on the project, a minimization of senior development time (oversight, quality control, etc.), and usually fewer quality control processes, which means the quality control processes are more reactive than proactive. To further reduce costs, it's recommended that you do whatever you can yourself, such as the design work and interface testing (we can show you how).
- Quality is of the utmost importance (healthcare software, e-commerce software dealing with transactions and sensitive consumer data, missile defense software, etc.): processes will be designed to ensure testing of the software in a formalized manner at all levels of the software development process. This will increase the costs, but it's worth it if the costs of errors downstream are not ethically and/or financially acceptable.

Implementation and Adaptation per Changing Needs & Constant Improvement:

At this point, your preferred process can be implemented to support your success criteria.

Just as software development itself is iterative and constantly improving, we believe the process should adapt based on your existing processes, stakeholder preferences, business goals, lessons learned, success criteria, and new or changed project considerations.



"Notice that the stiffest tree is most easily cracked, while the bamboo or willow survives by bending with the wind."

- Bruce Lee

We think it'd be a bit arrogant of us to think we have the best way.

You and your organization have learned a lot over the years, and we see no reason to change it if it's working. (If it's not, we will suggest methodologies.)

What matters to us is your happiness and the success of the project; how we get there is a component, and only just a component.

We adapt to your way of doing things, or we can provide [a way](#) to approach the project that works for us. How we approach it is per your preference.

Most things will go really well in a project, but there will be issues, questions, and concerns, which is entirely normal. In the ideal business relationship, we can talk through issues candidly and openly, fix the issues, and

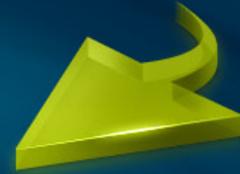
implement long-term process updates so that the issues are avoided in the future. (Related to this topic, we like the approach for a great vendor and customer relationship that's presented in Appendix B.)

A Baseline:

We do have a baseline software development process that we recommend to customers who don't have their own processes in place, or would like to learn what we use. We should mention that we adapt this to each customer's needs so what is presented here may be more than is needed for all projects.

"We think projects boil down to agreeing on what needs to be done, doing what needs to be done, and ensuring what was done works. This is by no means a linear process; it's iterative, full of communication, and adaptive."

- Ian McKilligan, CEO



We're often asked about our methodology; we prefer to think of it as a way of thinking.

Customer happiness is putting the user first.

We approach our development in an agile fashion, focusing on customer and user feedback and involvement, user interfaces, and user interaction, captured in the form of user stories and workflows. Development follows definition of what the user wants. Simply stated, we put the user first, and we don't consider a project successful until you do, which we think is when your users do.

Experience, earned through study and application.

We're not trying to be simplistic about this. We've been there. We have team members who've done project management for Fortune 50 companies, led and operated initiatives under ISO X/CMMI/ITIL/more standards, managed and performed user-based contextual inquiry and uncover/discovery, studied project management (PM) formally in college and via PMP (a candle's flame to the sun, considering the sun to be experience), and have used most PM tools out there, even having created some of our own, including a Kanban board and a "pure" PM, process-driven system.

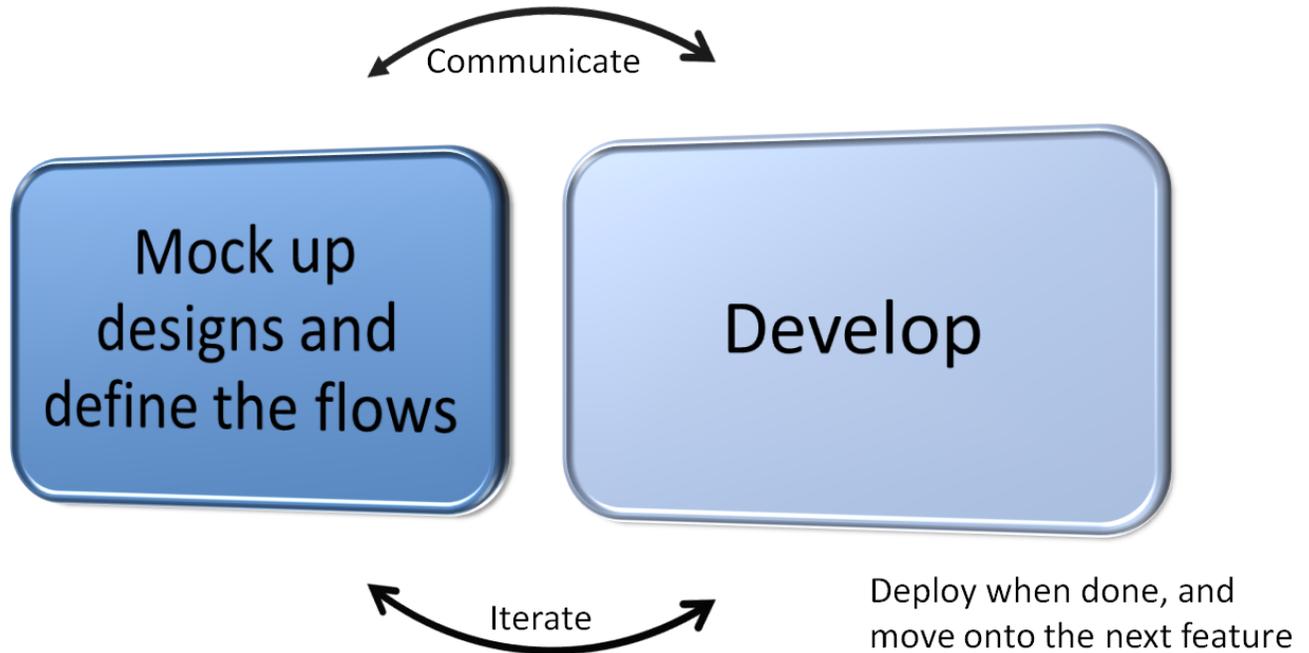
What doesn't good communication and accurately-focused action solve?

We think a project's success is directly proportional to the amount of good communication and good action that take place. This means talking about money, the realities of the project, and what's working or not in an open, transparent, and timely manner. This means acting efficiently after asking good questions to define issues and next steps.

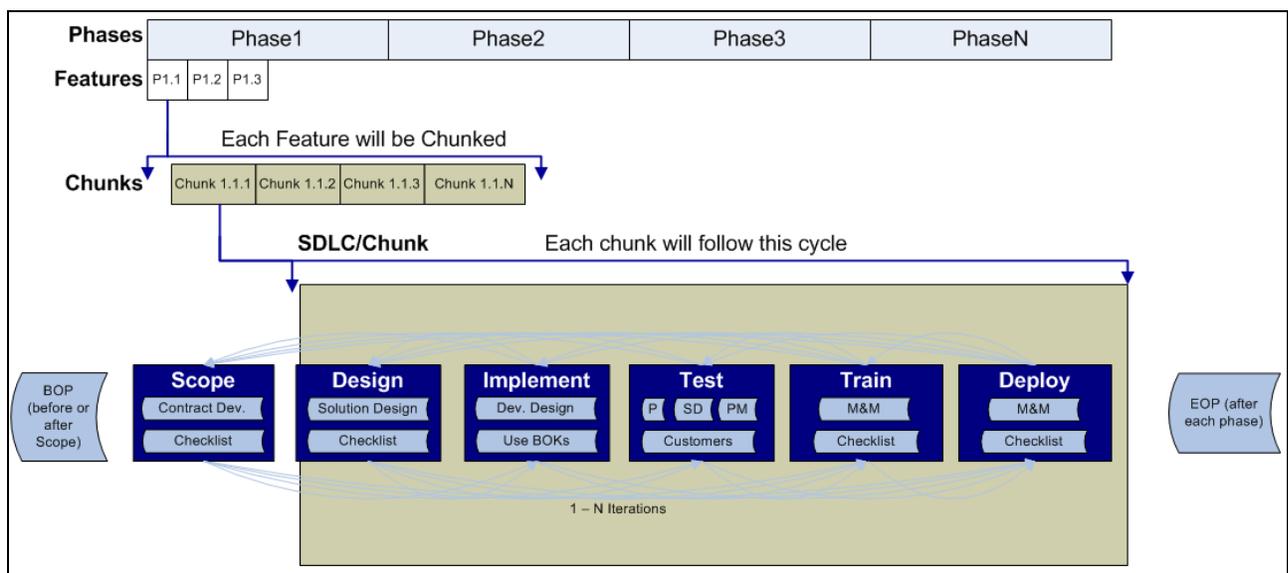


We'll first present a couple of workflows that we hope indicate broadly how we approach software development.

Software Development, Defined Simply



Software Development, One Way of Defining it Complexly



The following dives into more detail about some critical aspects of the process.

Know Where You're Going via Images / Pictures / Mock-ups / Wireframes / Low-fidelity User Interfaces ("mock-ups" from here on out): The "a picture is worth a thousand words" idiom applies here. The mechanism for this can be quite simple; that is, the medium does not matter. We and our customers have done these by hand on paper or white board, and then images were shared electronically by taking digital pictures of the results or by scanning them in. If you prefer to mock up the designs using software, anything that can draw will work, including software that most people have for creating presentations, documents, and custom graphics. For those that want software designed for the purpose, solutions such as Balsamiq, Azure, OmniGraffle, and Visio are popular ones. Please see Appendix A for example mock-ups.

Here are some notes on this important process:

- In terms of the value created for the time involved, we consider this step to be one of the most valuable in the software development process. It will save you time, it will reduce ambiguity, and it will save you money. We have familiarity with Business Requirement Documents (BRDs), and though useful, they can be replaced entirely by mock-ups. Focusing on one output reduces overhead by simplifying the process and focusing attention. As an example, in some cases, a mock-up can replace tens of pages in a BRD.
- Annotating the images using "call-outs" is really helpful. They can reference other documents if needed too. However, the goal should be to include all aspects of the business requirements in the interfaces themselves. This will save time in development and validation of the development results.
- Run the interfaces by representative users and colleagues to see if they intuitively make sense to them. Have them use the interfaces as if it was a real system. In most systems and in most cases, people don't seem to read the help guides so provide little to no help text and no introduction, and observe what the users can do on their own. To apply another idiom, "measuring twice and cutting once" sure saves a lot of money; that is, downstream changes are a lot more expensive than defining things well upfront. (This process could be considered as a form of "paper prototyping".)
- If you have final designs created, then we're past the point of creating mock-ups. If you haven't done the paper prototyping process though, you might consider it. Though you've invested in the high-fidelity user interfaces, it's still cheaper to change things now versus later.
- If you don't want to create mock-ups or don't have time to, that's no problem, we can do them based on your business requirements. If you prefer to not do this as part of the process, we understand; we really want to approach the work how you think it should be approached.
- Sometimes there are workflows that are not obvious by looking at the mock-ups. Here are some steps that can be taken at this point:
 - Creating a flowchart: any flowcharting approach will work. If you know a formal process for this, by all means, please put it to work. However, if you don't, that's ok; you're going to provide tremendous value and definition just by writing out the workflows however you want to. Googling "flowchart how to" or "flowchart examples" will help you to do this easily.

- Documenting user stories (simply tell the story of how the system will be used from the user's perspective, keeping in mind the user's specific role). Scott Ambler presents how to create and use user stories well here: <http://www.agilemodeling.com/artifacts/userStory.htm>. Here's another helpful resource: <http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/>
- How you organize and manage these mock-ups is your decision, and usually depends on the complexity of the system. We've seen people use a file structure, a ticket system, the modeling software itself, and a classification system. We prefer to use a ticket system so that everything that relates to a topic is organized within and easily accessible from that ticket.

Communicate with Developers via Functional Specifications: At this point, we will create functional specifications if needed, which are the specifications the developers use to implement the site. This serves the following purposes:

- It validates the mock-ups in that a detailed review of them is made, and detailed questions are asked if any ambiguity exists. This may cause more iterations of the mock-up process to happen, which is great because that'll save you money by correcting issues early in the process before actual development occurs.
- It produces documentation that developers will use to implement the site. Example outputs, which are produced as needed: 1) Entity Relationship Diagram – presents all of the database relationships, 2) Data Dictionary – documents all of the data fields and how they will be implemented, 3) Task Definition for developer allocation, 4) Test Cases, 5) Site Navigation Structure – navigational hierarchy of the site, 6) Additional Use Cases, 7) Class Diagrams – define static domain logic models, 8) State Diagrams – define operational state flows for the application, 9) Interface Boundary Interactions – demonstrate how different objects and/or third-party systems will interact with exposed APIs that our system would provide, 10) Architecture Design Layouts – demonstrate how software and hardware interactions and implementations will operate architecturally for the finished solution.

Start Building via Iterative Development: The development takes place. The Project Manager/Senior Developer (this person has a developer background, will be doing development on your project, is your direct contact, and is responsible for your happiness and the success of the project) will allocate most of the work to developers to perform, and then manage the process. Once completed by the developer, the work will be reviewed by the Project Manager/Senior Developer at the code level and at the user experience level to ensure that what was in specifications was delivered. Iterations will take place as needed. Once approved, the work will be released to you for your review. More iterations may take place based on your feedback. Once you approve it, it will be uploaded to the production system during the next scheduled release, at which point all of the new changes to the system will be tested again by the Project Manager/Senior Developer and you. (The structure above is cheaper and faster than using a single developer to do all the work, and it also allows for scalability and redundancy in your team, which is important for any system that will evolve and be supported over time.)

Ensure Customers and Development Team are In Sync via Communication and Expectation Setting: The processes above were presented in a simplified manner. Here are some of the other components that would be discussed and worked into the processes:

- Architecture considerations: assessing different options for infrastructure for the site based on usage and functional requirements
- Change request procedures
- Communication protocols
- Definition of testing procedures: unit testing, regression testing, interface-level testing
- Deployment considerations and process
- Development environment: definition, configuration, staging, and optimization (single-click deployments, for example)
- Documentation standards: code and user-level (help guides, context-specific help, etc.)
- Iteration definition
- Measuring success: before, during, afterwards qualitatively and quantitatively
- Release management definition
- Security design and considerations
- Stakeholder definition and responsibilities: escalation protocol, sign-off procedure, hierarchy, etc.
- Support plan
- Test plans for security: injection, cross-site scripting, taint-based checks, tests on operational code, etc.
- Ticket closure procedures
- Training plans: roles, mechanisms, validation, etc.
- User experience testing workflows and processes: personas, end user data gathering methods, user testing, usability testing, etc.
- Validation processes for markup
- Version control usage and technology

Next Steps:

Thank you for considering Up and Running's services. If you have questions, requests, or feedback, we'd be pleased to hear from you at any time. We hope to have further conversations with you, with the result being an approach that we're both comfortable with.

Thank you and Respectfully,

Ian McKilligan

Ian McKilligan
Up and Running Software, Inc.
Cell: 906-281-2627
Email: ian@upandrainingsoftware.com

Appendix A – Mock-up Examples:

Search Criteria

By Last Name [A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

Search

Filter by organization:

All contacts that have had activity with days.

All contats that have had NO activity within days and aren't closed.

Sales progression step:

Search Results

[First](#) [Prev](#) [1](#) [2](#) [3](#) [4](#) [5](#) [Next](#) [Last](#)

First Name	Last Name	Organization	Email	Phone	Sales Progress	Last Activity
Alina	Barns	CompanyX	alina@companyx.com	9999999	New Lead	08/24/2009
Aidan	Jones	CompanyX	aidanb@companyx.com	999999	Estimation	08/27/2009
Alex	Johnson	CompanyX	alexm@companyx.com	999999	Estimation	08/27/2009

[First](#) [Prev](#) [1](#) [2](#) [3](#) [4](#) [5](#) [Next](#) [Last](#)

Clicking on a letter brings up all contacts with a last name that starts with that letter. Note some characters appear as checkboxes, they should be their respective letters.

Number of results per page should default to 20. This should be controlled by a config variable.

When Advanced Search is clicked, the link goes away and the form reconfigures itself like this.

Column headers should be clickable to sort the current table of records.

Payment System

Name	Email	Rate/Salary	Hours	SubTotal	Adjustment	Final Total	Notes	Action
John Smith	John@aol.com	35	38	1225	<input type="text"/>	875		Send Success
Bob Roberts	Bob@aol.com	45	15	675	25	475	bonus	Send Employee doesn't exist
Nancy Jo	cy@aol.com	35	40	2200	-100	1000	reduced because of	Send

Summary: -\$1,564

Annotations:

- Clicking on the name, goes to the employee record
- Shown only when running in sandbox mode
- Clicking the edit icon will do a popup to add a note for the adjustment
- This field is computed automatically if subtotal or adjustment are changed. It can be manually modified too.
- Clicking send all, processes all employees that haven't been sent to yet. If an operation succeeds or fails, that counts as sending and won't be processed by send all.
- When a button is clicked, it's disabled until the operation finishes
- Results of the sending appear here. Either through a single Send click or if all are processed by Send All
- This is displayed in place of the green difference when the payment total is > than the payoneer balance
- The payment total and different are updated with javascript as the final total fields in the main list are adjusted
- This is retrieved by an API call to Payoneer

Peter Hanson
peter@upandringssoftware.com
 906-281-1178

Tags

VIP 

[Add Tag](#)

Info | **Activity** | Follow-ups | Sales Progression

Filter

From: To:

Type:

Keywords:

Filter legend tag is clickable to collapse the fieldset

- Activity
- [New](#) | [Collapse All](#) | [Expand All](#)
- Phone 01/15/2009 - Sync up meeting about DB - Files: 0
 Reviewed the quote and validated all assumptions. Need to update the estimate based on questions asked and answered
 Blah blah
 Blah blah
 File: [UAR Estimate - XYZ Corp - Drupal Site.xls](#)
 - Email 01/07/2009 - Database structure - Files: 1
 - Email 01/05/2009 - MOckups design - Files: 0
 - Email 01/04/2009 - Re: Database Design - Files: 1

By default, all activity is shown for the contact

Activity uses an accordion control, so you can quickly scroll through all activity. Clicking on a header expands/collapses that section. Multiple sections can be open at once.

Appendix B – Company & Customer PACT – How We Like to Work:

The COMPANY - CUSTOMER PACT

THE CHALLENGE

We, customers and companies alike, need to trust the people with whom we do business. Customers expect honest, straightforward interactions where their voices are heard. Companies work to inspire brand loyalty and deliver satisfaction while trying to understand their customers better. It is evident that we all have a crucial stake—and responsibility—in transforming the adversarial tone that too often dominates the customer experience.

A CALL FOR SHARED RESPONSIBILITY

Along with open, authentic communication comes the mutual responsibility to make it work. As each of us is both a customer *and* an employee, we share in the rewards and challenges of candor. By adopting these five practical measures, we can together realize a fundamental shift in our business relationships:

COMPANIES

1. Be human. Use a respectful, conversational voice, avoid scripts and *never* use corporate doublespeak.
2. Encourage employees to use their real names and use a personal touch.
3. Anticipate that problems will occur, and set clear, public expectations in advance for how you will address (and redress) issues.
4. Cultivate a public dialogue with customers so they feel they are being heard and to demonstrate your accountability.
5. Demonstrate your good intentions by speaking plainly, earnestly, and candidly with customers about problems that arise.

CUSTOMERS

- Be understanding. Show the respect and kindness to company reps that you'd like shown to you.
- Use your real identity, and foster your long-term reputation with the company.
- Recognize that problems will occur, and give companies the information and time required to competently address issues.
- Share issues directly, or through a forum where the company has an opportunity to respond, so it can work with you to solve problems.
- Give companies the benefit of the doubt, and be open to what they have to say.

OUR PACT

By working together in these ways, people build long-term relationships that lead to trust, strong communities, and sustainable businesses. We, as companies and customers, support this call for change.



Source: [SUPPORT THE PACT AT HTTP://CCPACT.COM](http://ccpact.com)