HGExperts.com
LEGAL EXPERTS DIRECTORY

CPR (Cooperative Project Recovery)
Reviving the Drowning Large-Scale IT Project (V2.0)
© Copyright 2007 by Warren S. Reid. All Rights Reserved

By: *Warren S. Reid,* Managing Director of WSR Consulting Group, LLC specializing in consulting and expert witness assistance in computer system turnaround, software projects, software failure, Internet and e-business failure, high-tech, and IT intellectual property matters. You can reach him by email wsreid@wsrcg.com and by phone (818) 986-8842.

According to the often industry-quoted Standish Group, *"Chaos Reports"*, software project failure rates are still unacceptably high — with **approximately 71% of all large-scale systems projects are either "challenged" (**meaning the projects are delivered well past the scheduled delivery date, materially over budgeted time and effort, and/or lacking critical features and requirements) — **or "scrapped"** before they are completed. Accordingly, every IT CXO and Project Manager must be aware of a turnaround process that can keep their sinking projects afloat and put them back on smoother, navigable waters.

"CPR (Cooperative Project Recovery): Reviving the Drowning Large-Scale IT Project" introduces my tested and proven four-tier, 13 step model which focuses on the actions that must be taken to turn around, overcome, and compensate for deficiencies in project management, methodology, technology, and people in live, ongoing projects — before you scrap the project, fire your staff, and/or move to litigation!

In a previous article entitled "Why Do Systems & Software Projects Fail? 2007" on the HGExperts site, I documented the 11 reasons why virtually all systems projects fail — it's been mostly the same over the last four decades, and it's the same all over the world — based upon my 36 years as a computer systems consultant and computer crisis consultant, including 20 years as a systems project turnaround specialist and an expert witness in software implementation failures in the United States, Canada, the Caribbean, Europe and Asia. Knowing why systems fail has helped me to develop and refine a system for project turnarounds which works. The 13 steps of the *CPR model* are briefly introduced below:

1. **Create a new project-wide *communication* system** that shares **honest, "not rosier than actual"** project status, issues, risks, recommendations and to-do follow ups to all relevant project members, executives, and other stakeholders, in a timely fashion and in a format the enables the responsible people to plan, act, proceed, and escalate issues as necessary. The old system obviously hid the real project status and progress through incompetence and CYA, or because no one could agree on what to measure and/or how to measure it.

2. **Think CSI — take the forensics approach.** Engage an independent expert to uncover, discover, dissect, and assess the root causes of the problems and issues that have hindered or crippled the project and how each of the parties' contributed to project woes. If the system appears to be full of bugs, was the data converted badly, or were the interfaces to legacy system poorly built? *Whose responsibility was that?* Did the scope of functionality change reasonably and through a proper change control process — or did scope change "stampede" through the project? *Whose responsibility was that?* Was the use of the promised SDLC methodology stopped because the customer didn't want to pay for all of the extra steps that were built in the methodology to enforce good quality, or because the vendor, integrator or outsourcer wanted to make up for cost and schedule overruns elsewhere on the project? *Whose responsibility was that?* If the system appears to be failing in the field test, is it the system, poor user training, or the customers' failure to make the business reengineering changes they promised to make? *Whose responsibility*

was that?

Conduct probing interviews at all levels, with all parties; review key project documents (internal and external) status reports and those "inconveniently truthful" emails that project members forgot they wrote; examine what steps in the methodology were ignored or given short-shrift; look at the tools that were used, and how they were used; explore the dynamics between and within the various teams; review quality standards; review staff qualifications.

3. **HUA! Define success!** Success means different things to different stakeholders. Developers want the "cleverest" code in the newest sexiest languages; users want an intuitive interface that will catch their errors before the system accepts them; cost conscious managers want the best solution for the cheapest price; accounting wants strict audit controls; marketing departments want funky customer features; IT may be striving for a bigger budget or to show the groups it serves that it understands their business needs and can get a job done on time and on budget; security wants to lock down any leaks in system access, etc. Success usually involves balancing and making tradeoffs between these various constituencies in terms of *costs, schedule, scope of features and functions, acceptable risks, stakeholder expectations, and the quality of the system.* Add to that concerns about portability, testability, reliability, usability, and maintainability for good measure, and *you get the picture*!

   Spelling out success criteria in writing for the new project turnaround is difficult indeed — especially with so many different interests at stake. If you don't decide as a collective NOW what is meant by "success," then you'll never know if you're going in the right direction. Wouldn't you rather know what to target now?

4. **Revise your tasks and estimates; revise your contract.** Get agreement. Is success as envisioned at the project's outset still feasible given where the project is today? Does the business case still make sense? Does the ROI and cost/benefit still please the corporate decision-makers? Is this the best use of IT and department resources at this time for the company? Do you need team-building exercises? Is the new team confident? Empowered? Appropriately skilled? What about the vendor's team? The systems integrator? The outsourcer? Your management? Your users? Your steering committee?

   Identify, weigh, and cost your risks. Identify risk trigger points. Plan mitigation strategies and reporting metrics to track the potential onset of risks. Consider now-known and likely risks as you go forward. Virtually all successful projects plan for: project resource and staff turnover; the hidden complexity of data conversion/migration and legacy interfaces; misunderstood requirements; unclear roles and leadership; unclear/uninformed methodologies; lack of automated tools; non-use or misuse of the critical path method and earned values; inexperienced staff; competition and regulatory changes; technology disruptors, and more.

5. **Decision Point: Get On! Get Off! Get Away! Get Down!** Now that you have: (a) reviewed, uncovered and analyzed what happened to this project, and discovered the root causes of the problems and issues that caused the project/system to spin out-of-control, and (b) re-visited and re-confirmed the business need and business case for the system from perspective of economic, organizational, operational, and/or executive support needs, and (c) identified in writing the acceptance/success criteria for the system and project, and (d) identified the risks associated with moving forward with the team(s) and the project(s), you are ready to make a decision – or some would say, you've arrived at the "**tipping point**." You and the organization's leadership must make a decision about whether to go forward with the project, and if so how best to do that. Some questions to consider on whether to proceed: Does the system still make economic sense? Is it better to look at other alternatives? Do you start a different project that has more priority in the current circumstances, and defer this one for later? Do you re-scope the project? Is only a piece of it needed now? Other questions to consider on how to proceed: Do you go ahead with the same players, same resources, in the same roles, and continue to complete the originally scoped, but now failed, system? Can we continue to work with the selected vendors and suppliers and integrators? Are they willing to negotiate a reasonable contract that continues to give appropriate incentives to all parties to continue to perform cooperatively to get the job done? Should you outsource some of the work?

One decision may be to *get on* i.e. proceed with the project with the same players. Another alternative might be to *get off*, i.e. proceed with the project but with different players. Another alternative is to *get away*, i.e., walk away from this project altogether, and commit any new projects with the CPR method from step 1 — because almost all of the concepts in CPR work for new projects that you want to go successfully — as well as for turning around runaway projects in trouble. Finally, you, or a third party (vendor, integrator, developer, outsourcer, implementation consultant, etc.) may *get down*, i.e., decide to sue one another. Each of these four choices involves following the CPR methodology to its conclusion, including the start of new projects, or proceeding with dispute resolution. As it happens, the analysis needed to determine how each party contributed to a failed project for purposes of dispute resolution and the analysis needed determine strengths and weaknesses in a case is the same as the steps defined in the CPR methodology. In that sense the CPR method has universal application.

6. **Focus on the "Right Stuff" requirements and priorities**. Determine what are the "right" requirements — that is the minimal acceptable requirements now — with which you can live. Forget a "big bang" approach, you already blew it, and this time you need some early and frequent successes for and by the team. Determine and stay focused on the "true" priorities. Develop and change project management procedure and organization and reporting, and stick to it if it works. Create and enforce an escalation process to handle scope changes that threaten to throw off focus on getting the project done. Rebuild the relationships within the project teams and across all of the stakeholders. Identify, acquire, and train on the right automated tools. Achieve team buy-in!

7. **Put the "Right" people and team in place!** Arguably the Project Director/Manager job is the most critical role for success. The most successful project managers know the technology, have experience in the industry in which the system is being deployed, are team players, and most importantly, are inspiring leaders. Often, at least one Project Manager is let go as a way to show that the reconfiguration and rejuvenation of the project is *for real*. If there is bad blood in the upper levels of project leadership, then someone has to be cut. If someone new is brought in, then s/he must hit the new ground running – there is no time for on-the-job training in this go-round. Substitutes for key personnel must be negotiated before restarting the project. And remember, the relationship between staffing and scheduling is not linear; reducing qualified staff by 15%, will likely increase project schedule and duration by **much more** than 15%.

Now what about the "panacea" almost everyone asks about — **Outsourcing — especially to India.** While this may seem out of place in a systems project turnaround methodology, outsourcing IT and/or IT projects will almost always be raised as an alternative to get non-outsourced projects back in track. It is critical to keep in mind that up to **59% of all outsourcing contracts fail** (per Compass Consulting , Forrester , and A. T. Kearney studies among others). Yet many have been quite successful at it — although surprisingly very few in the 1st year of an arrangement. Understanding country and business culture, the new middle-classes, the country's infrastructure, the hardships vs. the promised benefits time zone differences bring, and understanding the country's security/privacy laws, as well as the all-important commitment to deploy essential managers from the customer's staff to the outsourcer's site (and sometimes vice-versa) are all critical for creating and managing a successful outsourcing arrangement. It is worth remembering that the anticipated and often touted 60-70% savings from outsourcing non-trivial software development **rarely** materializes. Instead, a 20-30% total savings after the first year appears to be a more realistic number for those arrangements that are considered successful.

8. **SWAT — SoftWare Adjudication Team.** The SWAT is commissioned to head off project issues while they are still molehills — and not let them become enormous obstacles to project success. The SWAT should consist of: (1) the independent consulting expert (who assessed the project status in Step 2 above) or an expert witness in the field; (2) an IT or industry professor in the application area; and (3) an attorney/mediator who KNOWS information technology. Add one decision maker from each party. The *SWAT meets regularly*, typically semi-monthly, whether or not issues are brought up, and resolve issues so that smaller ones don't add up to create a big problem, in order that the project continues forward, and so that the various project managers and stakeholders do not have

to fight it out among themselves. SWAT decisions and recommendations may or may not be binding — but they must always be well heard and understood!
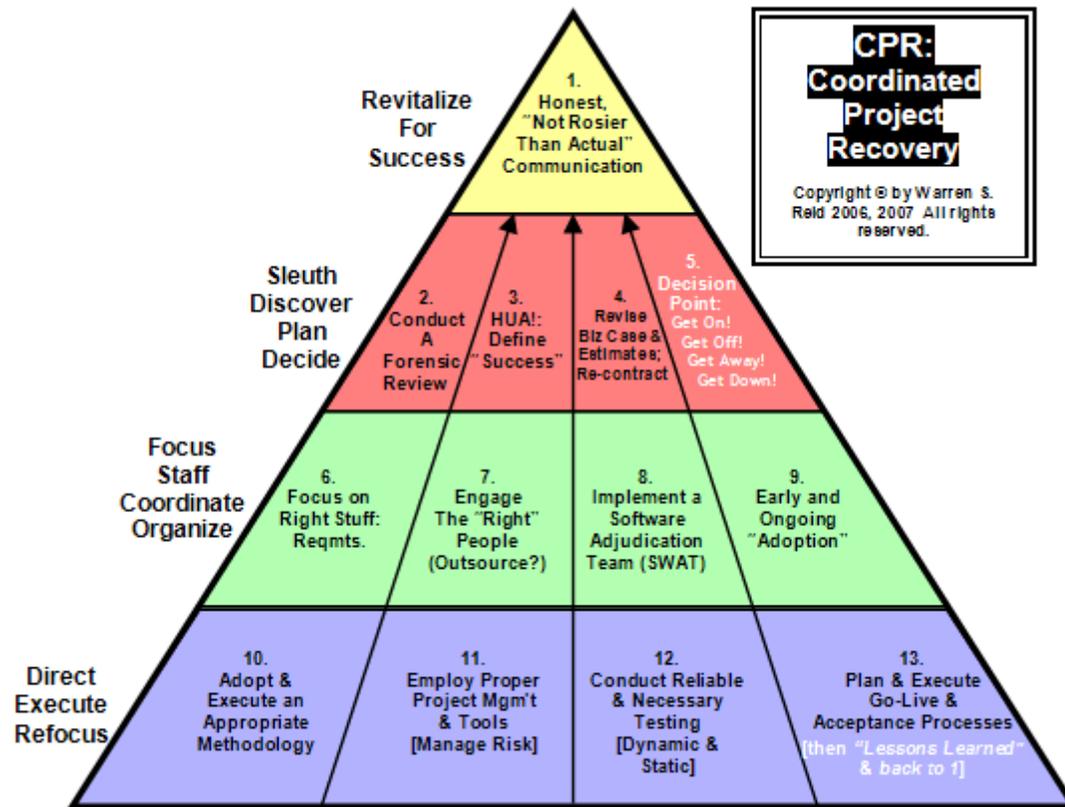
9. **Move to "Early & Ongoing Adoption" of modules and application components** — *i.e.*, **start thinking LESS is MORE**. Break down the project into smaller module/sub-system deliverables, each with a work plan of just a few months. Achieving quick milestones and early and ongoing project victories will give the users, stakeholders, and the project team the success mentality, satisfaction, and respect they need to regenerate and sustain their enthusiasm and commitment. In addition, smaller deliverable sub-systems are exponentially easier to manage and allow the team and users to test and refine their own help deck procedures, training environments and practices, testing regimen, methodology adherence, and maintenance procedures before full rollout. In addition, constituents get to see what they are paying for early on, instead of waiting for one, two, or more years to see big bang results.

10. **Outsourcing (especially to India).** While this may seem out of place in a systems project turnaround methodology, the fact is that outsourcing IT and/or IT projects will almost always be raised as an alternative to get a project back in track. It is critical to keep in mind that up to *59% of all outsourcing contracts fail* (per Compass Consulting, Forrester, and A. T. Kearney studies among others). Yet many have been quite successful at it — although surprisingly very few in the 1st year of an arrangement. Understanding country and business culture, the new middle-classes, the country's infrastructure, the hardships vs. the promised benefits time zone differences bring, and understanding the country's security/privacy laws, as well as the all-important commitment to deploy essential managers from the customer's staff to the outsourcer's site are all critical for a successful outsourcing arrangement. Also true, but beyond the scope of this article, is the empirical evidence and experience that the anticipated and often touted 60-70% savings from outsourcing non-trivial software development **almost never** materializes — a 20-30% total savings *after the first yea*r appears to be a more realistic number for those arrangements considered successful.

11. **"All *GOOD* methodologies are the same!"** is my most misquoted statement. As an example of project management methodology, all the good ones have the same basics: understand/elicit requirements and scope; create an agreed upon project charter; create a work breakdown structure (a detailed work plan); identify predecessor and successor tasks; decide on and assign proper staff; estimate time and schedule; set up the project team organization structure; capture actual time and compare it to estimated time; identify "earned value" and estimates to complete; identify late starting tasks; calculate and determine the reasons for variances; create and update your critical path, etc. Note that PMBOK, SWEEBOK, ISO-9000, Mil-Spec Standards, DOD standards, commercial methodologies, and most Fortune 1000 internally-developed methodologies all include the same tasks — but they are also, at the same time, different. The emphasis here is on **"good methodologies"** — and you have to be sure to select one that is a match with your application, your industry, the criticality of the software to the developer group and the user group, your organization/company style, and understood by your staff and organization. And yes, YOU MUST ACTUALLY FOLLOW IT! When you deviate from the methodology, and **at times you will have to**, you must first assess, document, communicate, and accept the risks — and do so for a good reason!

12. **Employ Proper Project Management & Tools.** Research, plan for, select, train for, and implement any of the many efficient and effective automated tools. There are tools available for many important repetitive processes including tools to help you estimate, develop work breakdown structures, report project progress, determine "earned value", smooth staffing levels, chart your "critical path", manage scope changes. Others create and manage code changes, test scripts, test data and error/defect logs. Yet others create documentation, help create computer based training sessions, and more. Provide a contingency in your schedule to accommodate a learning curve and the fact that some of the newer or less robust tools may introduce errors of their own. Also, provide for the cost to license, train, and integrate the tool into your project, as this can be expensive and time consuming. Therefore, determine your tipping point and move ahead accordingly.

13. **In the Real Estate business it's "location, location, location!"** In large-scale projects, it's **testing, testing, testing!** This can not be overemphasized! With all the levels and

types of testing **accounting for more than 50% of all of the time required for large development projects** and/or large implementation projects (and especially ERP implementations where the percent is even higher), testing must NOT be given short-shrift, poorly planned or casually executed. "Testing," as I use it here, includes among other things, the tasks of: developing test strategy; planning the tests; setting up a secure test environment; identifying the proper test team; creating test scenarios and cases and performing the myriad of different kinds and levels of tests; documenting results; debugging, correcting, retesting, and regression testing and acceptance; and performing related project quality assurance and quality control functions. Focus your testing on the right **areas**, at the right **levels of detail**, at the right **time**, with the right **people**, using the right **tools**, and in the right **environmen**t. Do your testing early and do it often! *Do not cut corners on the User Acceptance Tests* — doing so almost always results in project/production failure.

14. **Go Live Readiness and Go Live.** OK, you've followed good methodology and SDLC practices, and you've tested, QA'd and provided solid change management, realistic assumptions, and have defined "Success." Now it's the moment of truth. Before reaching that point you should answer the question, "Are you ready to go-live?" How many errors are acceptable and what level of functionality is acceptable **for now**? Are the users trained, the network and hardware deployed, and the security codes/systems set up? Do the users need refresher training? And what about your help desk? Is the testing completed? Is your maintenance group (the folks that will inherit the fixing of the system) knowledgeable about the system and how it was architected, designed, programmed, tested, configured, customized, and implemented? Have they inherited the proper tools and test suites from the development group? Are the backup and recovery/restart procedures in place? Like a successful home builder, successful projects and project professionals have punch lists of readiness criteria in place and check each item off before going live. This allows the project managers and steering committee to assess and communicate the risks of open items, and, if they have to move forward, to do so with their eyes open and mitigation plans in the ready.

    After go-live, there will still be a need to address some problems, mistakes and misconceptions during the first months of conversion, implementation and production — although far fewer unanticipated and time consuming problems when CPR method has been followed. It is now time for the team(s) to draft up a "Lessons Learned" document. This very important project document itemizes both the good and the bad lessons learned from the instant project for the benefit of future projects and project teams — and as such needs to be catalogued, circulated, and placed in a secure but accessible repository. When properly used, these documents can help future teams, executives, project managers, contract managers and attorneys duplicate the successes and avoid/address/mitigate, as appropriate, the bad situations and decisions identified in lessons learned in a non-disruptive way so that the project will successfully complete.

Note that the CPR model, in many ways, follows many of the tried-and-true management processes and doctrines employed in non-systems project arenas. Each of the 13 steps tie to management principles such as Planning, Discovering, Staffing, Organizing, Coordinating, Directing, Executing, Refocusing, etc. It's no wonder Baseline Magazine recently embraced the model in its November 2006 issue in "The Project Management Center" section.

CPR: Coordinated Project Recovery

Copyright © by Warren S. Reid 2006, 2007. All rights reserved.

Revitalize For Success
1. Honest, "Not Rosier Than Actual" Communication

Sleuth Discover Plan Decide
2. Conduct A Forensic Review
3. HUA!: Define "Success"
4. Revise Biz Case & Estimates; Re-contract
5. Decision Point: Get On! Get Off! Get Away! Get Down!

Focus Staff Coordinate Organize
6. Focus on Right Stuff: Reqmts.
7. Engage The "Right" People (Outsource?)
8. Implement a Software Adjudication Team (SWAT)
9. Early and Ongoing "Adoption"

Direct Execute Refocus
10. Adopt & Execute an Appropriate Methodology
11. Employ Proper Project Mgm't & Tools [Manage Risk]
12. Conduct Reliable & Necessary Testing [Dynamic & Static]
13. Plan & Execute Go-Live & Acceptance Processes [then "Lessons Learned" & back to 1]

References:

1. See Standish Group "**Chaos Reports**"
2. **HUA** = Heard, Understood, Acknowledged!
3. Outsourcing Contracts **Compass Consulting**, June 26, 2005
4. Captive Offshoring Centers Are Imploding **Forrester**, May 1, 2007
5. Offshore Success Is Uneven, Says Kearney Study - **A. T. Kearney**, May 9, 2007
6. **The Project Management Center** - Baseline Magazine, November 2006

Disclaimer: This updated version (V2.0) of **CPR (Cooperative Project Recovery): Reviving the Drowning Large-Scale IT Project (Version 2.0)** was developed by Warren S. Reid on August 15, 2007. The information contained on the http://www.wsrcg.com/ website and in this article is for education purposes and to stimulate thought, discussion and even controversy. Each real life situation is different! Accordingly, every solution must be tailored or heavily modified to the specific situation at hand. Please contact Warren S. Reid, Managing Director 818-986-8842 or email: wsreid@wsrcg.com to discuss the contents of this article further and how WSR Consulting Group, LLC might help in particular circumstances, or seek other professional legal, industry, and/or expert consulting advice as applicable **before** using any of the ideas and materials contained herein.

More on Why Do Systems & Software Projects Fail? 2007 and Computer & Systems Failure Litigation