***[TopCoder Direct]***

*Conceptualization Phase*

Conceptualization Document

Template v 1.0

May 20 2010

Metric Analysis

Table of Contents

1.0 Introduction 3

1.1 Overview of Application 3

1.2 Project Objectives 3

1.3 Assumptions 4

1.4 Limitations 4

1.5 Open Items and Risks 4

2.0 Existing Business Flow 6

2.1 Overview 6

2.2 Context Diagram 6

2.3 High Level Workflow 7

2.4 Dependencies 8

2.5 Current Limitations/Problems 9

3.0 Proposed Workflow 10

3.1 Overview 10

3.2 Context Diagram 18

3.3 High Level Workflow 19

3.4 Use Case Diagrams 22

4.0 Business Requirements 27

4.1 Users 27

4.2 Inputs/Outputs 27

4.3 Dependencies 28

4.4 Task Requirements 29

4.5 Security Requirements 34

4.6 Performance Requirements 35

4.7 Data Migration 35

* **Introduction**
* **Overview of Application**

Currently TopCoder offers Co-Pilot opportunities to member. Co-Pilot responsibility is assisting Project Manager in contest management. One of drawback in TopCoder process is the fact that usually no member is following the project during whole project life. Co-Pilot is the only member that will work on project from beginning to end.

Current Co-Pilot process is very simple and not as efficient as it can be. Co-Pilot is selected by Project Manager, no statistics is available, Co-Pilot responsibilities and payments are not adjusted optimally.

To improve the quality and responsibility of Co-Pilots and at the same time make their work more comfortable and productive, several changes will be applied to Co-Pilot mechanisms.

Improvements to current process will affect following areas:

* Co-Pilot payments;
* Co-Pilot selection;
* Co-Pilot reliability and quality of work history;
* More tight Co-Pilot and client interactions;

New system will restrict access to Co-Pilot opportunities for members of Co-Pilot pool.

The application will be based on existing TopCoder applications:

* TopCoder Direct for customer usage;
* Online Review for Co-Pilot submissions (based on these submissions Co-Pilot will be selected);
* TopCoder Web-Site for viewing Co-Pilots profiles and statistics;
* TopCoder Forums for Customer and Co-Pilots communication;
* Jira for BugTracking, VM assignment.

New process will make Co-Pilot selection more similar to other tracks. A contest will be run to determine Co-Pilot for the project. Not only submissions but Co-Pilot profiles will be analyzed by client on selection stage. Payment model will be changed in order to reflect Co-Pilot work quality better.

Also, some convenience tools will be created to make Co-Pilot work easier.

ROI Metrics

Though ROI metrics cannot be completely defined, but some thoughts for the return of investment can be provided.

* New system will highly increase responsibility of Co-Pilot. Having public profile and project history open for clients, Co-Pilots will likely be more responsible.
* New payment model will provide higher payments for better Co-Pilots, at the same time lowering cost for client and increasing TopCoder income.
* Creating projects only with Co-Pilots without a lot of help from PM will allow taking more projects than now without hiring more fulltime employees. It is much more expensive to hire full-time PM than having remote Co-Pilot because of other taxes, insurance, work place, office rent and equipment costs.
* Having Co-Pilots profiles and better structured initial documentation, more detailed review and analyzes of finished project can be done. It will be easier to determine what errors were made during any phase.
* **Project Objectives**

The business objectives of this project are listed below. Delivering these objectives will deliver the expected benefits of the application.

* Make Co-Pilot work more structured and responsible
* Make Co-Pilot selection process more objective
* Make Co-Pilot history and profiles available
* Improve Co-Pilot and Client communication process
* Reduce TopCoder Management Team workload on projects with Co-Pilot involved
* Make Co-Pilot earnings better reflect the skills and efforts made by Co-Pilot
* Restrict Co-Pilot opportunities to a pool of well-skilled TopCoder members
* Give client an ability to choose Co-Pilot based on presented documentation and Co-Pilot profile
* Provide client a way to see and explore Co-Pilot work history
* Provide client a way to write a feedback for the Co-Pilot
* **Assumptions**

Some assumptions generally must be made in order to write a succinct definition of the application. Some assumptions are technical (e.g. “the new system will employ a normalized database schema”), while others are business natured (e.g. “the client will provide an enterprise Oracle environment”). Assumptions critical to the success of this project are listed below:

* New system will be based on existing TopCoder applications
* New system will be implemented in Java, as all TopCoder applications are written using this language/technology.
* Initial Co-Pilot pool will be filled manually.
* Project can run with PM without Co-Pilot
* Project can run without PM but with Co-Pilot
* Project can run both with PM and Co-Pilot
* Submission screening will be performed by TopCoder managers currently
* Existing Co-Pilot statistics will be moved to new system.
* Client will be using simplified interface and will NOT need to use OnlineReview, Jira, Wiki. Only TopCoder Direct, Forums, and the website (for profile viewing) will be used.
* **Limitations**

Every project has limitations on the scope of the problem it attempts to solve, on the capabilities of the implementation technology, etc. Some limitations are assumptions on the extent of feature scope. Others are restrictions on resources or methods for achieving the objectives. The limitations of this project are listed below:

* The access to Co-Pilot pool will be limited to high-rated/reliable members.
* There will be no second place in Co-Pilot contests
* Only new projects will use new system
* **Open Items and Risks**

During the conceptualization of the project and in the process of writing this document, some issues may have been discovered, and others may remain open.

* The payment model is subject to change. At least percentages can be tuned
* The content of contest status page is subject to change.
* Proof on concept can’t be made during some testing. Only real projects can be a proof. For this application the quality of new payment system can be determined only after launching the platform and having some projects done.
* If a lot of projects will arrive to TopCoder, pool will have to be extended and less experienced members will become Co-Pilots.
* The process of ‘changing Co-Pilot’ in the middle of the project is not well-defined. It must not happen normally, the Co-Pilot is not allowed to leave project. However, this is a possible scenario (Co-Pilot leaves the project).
* The rules of forming ‘Problems’ and ‘Attentions’ on contest status page are subject to change.
* The set of questions in initial client documentation can be changed
* The set of questions for client review process can be changed
* The description of Co-Pilot system for the client is not created
* There will be no appeals in Co-Pilot contests

* **Existing Business Flow**

If this is an entirely new application, this section should be completed only if there is an existing business process.

* **Overview**

This section documents the current business process. The purpose of doing so is to confirm the author understands the current procedures or opportunity. This will provide a backdrop against which the benefits of the new proposal can be illustrated in the next section.

Currently, TopCoder provides a distinct way for the members to work. It is Co-Pilot opportunity. Co-Pilot is a member who helps PM with project during all project lifetime.

Co-Pilot payments are fixed fees for contests run.

Current Co-Pilot responsibilities are:

1. Creation of contest specs and management of spec reviews.

2. Setup and launch contests

3. Monitor/Manage contests as they are in progress, including questions in the forums, issuing VM's, etc.

4. Testing the final submissions

5. Optionally, merging the final submission into the main application branch and validating that the code works on a VM. In other cases this is done as deployment task.

6. Working with the customer team to meet deployment schedules.

7. Run BugRaces and check their results to fix errors in the application/documentation.

Co-Pilot opportunities are posted as Bugs in Jira. PM chooses one of Co-Pilots from voted members. There are no formal rules for Co-Pilot selection currently; PM selects Co-Pilot at his own discretion.

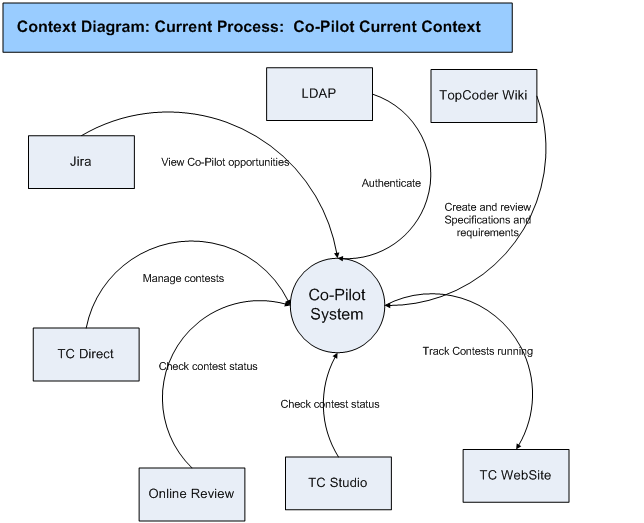
After Co-Pilot selection, Co-Pilot is responsible for creating Game Plan. Game plan is a project plan adapted to TopCoder software production model.

This project plan is continuously updated reflecting changes after some key competitions (Conceptualization, System Architecture).

For each finished contest Co-Pilot earns fixed percent of contest prize. This payment is distributed in following way: 60% is paid at the contest finish, another 40% - at the project completion.

* **Context Diagram**

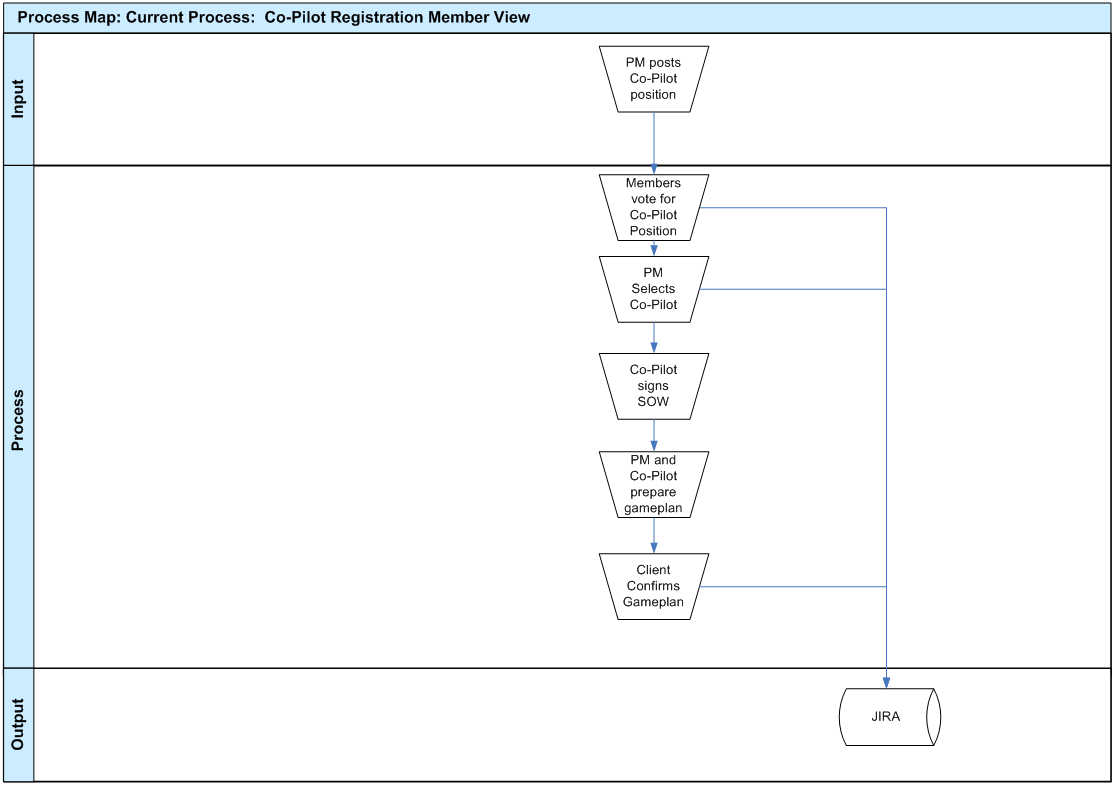
The Context Diagram illustrates the systems, modules, business processes, etc that feed or interact with this application, where the application is the “center of the universe.”



* **High Level Workflow**

The workflow required to complete the primary objectives of this application is described below. The workflow is business-centered, and includes “decision forks” for decisions the business user, or application, must make to achieve the objective. The workflow omits application faults or exceptions. Individual tasks in the workflow are briefly described in section 2.3.2.

* *Workflow Diagram/Process Map*



* *Workflow Description*
* PM posts position at page <http://www.topcoder.com/wiki/display/tc/Active+Copilot+Opportunities>
* Problems: This approach misses the overall contest-based TopCoder development style. It is not as useful as other tracks are.
* Then members can vote for the position
* Problems: there’s no strict timeline for the ‘contest’. No information except for interest in work is provided by voters.
* After that, Co-Pilot will be selected by PM.
* Problems: The selection can be quite “random” and will not reflect real project needs
* Co-Pilot signs SOW.
* Problems: none
* Currently, Co-Pilots can be responsible both for overall project and for some subset of competitions. In this case, no Game Plan creation is needed.
* No problems here
* If GamePlan creation is required, Client must confirm it before moving project forward.
* Problems: Game Plan at this stage is like a guess. No communication with client was done before this.
* After client confirmation, Co-Pilot will create and manage contest.
* No problems here
* **Dependencies**

The list below enumerates external applications, business procedures, personnel, etc, that are required to achieve the objectives of the current application. For example, if the application is a data warehouse, it will depend on one or more OLTP systems at the enterprise. Those systems would be listed here.

* TopCoder JIRA Bugtracker
* It is used for Co-Pilot opportunity posting and bug races run by the Co-Pilot
* TopCoder forums.
* Forums are used for communication between Co-Pilot, PM, Client and Contestants
* TopCoder Direct
* Is used by client to initiate the project and by Co-Pilot for contest creation.
* TopCoder Online Review
* It is used for contest status check and contest review results analyze.
* TopCoder website
* It is used for overall information getting, checking registered member profiles and watching contest statuses.
* TopCoder wiki
* It is used for Specifications and Requirements creation and viewing
* **Current Limitations/Problems**

Limitations of and problems with the current application are listed below. These are the issues that the new application strives to address.

* There’s no Co-Pilot profile available
* There’s no project history for Co-Pilot
* There’s no formalized process for Co-Pilot selection
* It is hard to create contest with Co-Pilot and without PM
* Co-Pilot payments do not reflect the skills and efforts made by the Co-Pilot
* There’s a chance for not very skilled member to become a Co-Pilot
* There’s no defined penalization for failed contests and projects
* Co-Pilot has no direct control over project timeline. It is manager responsibility.
* A lot of communication is done over e-mail, so it is hard to keep track on all contests.
* There’s no page where Co-Pilot is able to see and track all running contests.
* There’s no page where Co-Pilot can see current detailed timeline

* **Proposed Workflow**
* **Overview**

This section includes a concise description of the features and operation of the proposed solution. It also compares and contrasts the solution with the current process. The context and workflow of the proposed solution are defined in subsequent sections.

All these sections add new functionality and solve some of previous version limitations. All they are ‘contrast’ with previous solution.

* *Client Initial documentation*

To start the contest client should provide some initial documentation. This documentation will be similar to conceptualization questionnaire, but will have some other questions in it. Any additional documentation that client can provide is desirable.

The preliminary list of questions:

Only for existing system upgrade

* Short description of current application.
* Who uses the application currently?
* What is wrong with current application? Why should it be changed?
* What dependencies exist for the current system?
* What technical details can you provide about existing app? (Environment, etc.)

For existing system upgrade and new application

* What are you expecting to achieve in new application?
* Who will use new application?
* What systems will depend on new application?
* What systems does new application depend on?
* What is expected deadline for the application production?
* What technical details can you provide about new application requirement? (Environment, etc.)
* What are future directions for the system?
* What type of the application will it be (these will be checkboxes, as complex system can consist of several applications):
* Web-application
* Mobile application (define platforms: iPhone, Android, Blackberry, etc.)
* Desktop application (define platforms: Windows, Linux, Mac OS)
* Web-Service
* *Client Communication Tool*

Currently, client is communicating with competitors on forums. It is convenient for contest participants, but can be not very convenient for client. It will be easier for client to answer using e-mails. Similar approach is used in Google groups (<http://groups.google.com/>). In this system user can answer to special ‘group email’. After that, message will be automatically posted to the forums. Similar tool can be created for client communication on new Co-Pilot platform.

For each forum new email must be created automatically (i.e. [elephant-project-co-pilot@topcoder.com](mailto:elephant-project-co-pilot@topcoder.com)). Writing a mail to this address will cause a new thread to appear. Responding to a message from this mail will cause posting to existing thread.

Currently this tool is a suggestion and the tool will potentially be created later, after the new process run. It will be described better after more contests run and more client feedbacks gathered.

* *Client selection*

Before starting project, client may hire Co-Pilot to help with the project. The description on what a co-pilot is must be provided to the client. The information must contain:

* What benefits will the client obtain: experienced TC member, assist with contest creation, question answering and so on.
* How much will it cost to the client.

If client wants to have Co-Pilot, a contest will be created for Co-Pilot selection.

Client will select one of submitted Co-Pilots based on submission and Co-Pilot profile. If client does not like any submission, he is able to discard all Co-Pilot inquiries.

The client can also hire a TopCoder Project Manager for his project. This will cost him additional money, but will make the process better managed and predictable.

During the contest the client will communicate with Co-Pilot on forums and/or using e-mail based communication tool.

The client will not be required to use online review, jira or any other TC systems except TC Direct and TC Forums.

* *Co-Pilot Pool*

Only members from Co-Pilot pool will be allowed to take Co-Pilots positions. Initially Co-Pilot pool will be formed as follows:

* Members who have previous successful Co-Pilot work history;
* Members who have rating > 1200 in architecture, specification or concept;
* Members who have reliability 100% in architecture, specification or concept and 3 or more contests with 1st or 2nd place in these tracks;

Requests for Co-Pilot pool entering will be handled by TopCoder project managers manually at this moment.

* *Co-Pilot Profile*

New tab will be added to TopCoder profile page. It will contain information about Co-Pilot work history.

Following information will be presented initially. More metrics can be added later.

|  |  |
| --- | --- |
| **Name** | **Description** |
| Number of projects | Number of projects run. Clickable, leads to project history page. |
| Total number of contests. | Number of contests from all projects. Includes number of contests that succeeded (if the contest was reposted, it is counted only once) |
| Total number of reposts | The number of contest reposts without changing prize/scope of the contest |
| Total number of failures | The number of contest failures. Failed contest is defined as contest, which either was not reposted at all or prize/scope for it was changed after repost. |
| Active/Suspended | Says if Co-Pilot account is active or suspended due to reliability issues. If account is suspended, the activation date must be shown. See 3.1.7 for suspension rules. |
| Number of suspensions | Shows count of suspensions. Must be 0 for good Co-Pilot. See 3.1.7 for suspension rules. |
| Earnings | The amount of money earned as Co-Pilot. This will be hidden if member decided to hide. |
| Total number of bugraces | Number of bug races in all projects |
| Number of current projects | Number of projects in which Co-Pilot is involved currently. The expected project ends must be available here. |

Project history page will show the list of projects run by the Co-Pilot. The real names of the projects must be hidden. Following information must be shown for each project:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Planned number of contests. | Number of contests planned by Game Plan. |
| Real number of contests. | Number of contests that run in reality. |
| Number of reposts | The number of contest reposts without changing prize/scope of the contest |
| Number of failures | The number of contest failures. Failed contest is defined as contest, which either was not reposted at all or prize/scope for it was changed after repost. |
| Planned duration | Shown planned project duration according to Game Plan |
| Real duration | Shown planned project duration. |
| Planned number of bugraces | Planned number of bugraces according to Game Plan |
| Real number of bugraces | Shows real number of bugraces required. |
| Customer feedback | This will be a feedback from customer. It will be pre-moderated by TopCoder PM and contain some textual information about Co-Pilot work. |
| PM Feedback | This will be a feedback from PM. |

Also special page will be created to see statistics grouped by contest type. Following information will be available for each contest type:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Planned number of contests. | Number of contests planned by Game Plan. |
| Real number of contests. | Number of contests that run in reality. |
| Number of reposts | The number of contest reposts without changing prize/scope of the contest |
| Number of failures | The number of contest failures. Failed contest is defined as contest, which either was not reposted at all or prize/scope for it was changed after repost. |

Initial ratings will be filled from existing project\contest statistics. Potentially, some manual work will be required. In cases, when there’s no enough information, the feedback from PM must be filled with more details.

* *Co-Pilot Selection*

In contrast with existing approach, new Co-Pilot system will be more objective to the Co-Pilot selection process. It will be a kind of mini-contest, where each member will be required to submit Game Plan and Development Strategy document. Member will be able to unregister from the competition during registration phase. This will allow potential Co-Pilot to get more information about the project from forums and decide if their skills match the required project.

After submission, short screening phase will be started. The screening will be performed by TopCoder Managers currently, but in future, review board can be created. Screening will have a list of questions, that checks:

* Game Plan
* Development Strategy
* The fact that these two documents are correlated.
* Grammar and spelling of the documents will be checked, but some errors here will not lead to screening failure.

After that, client will decide, which submission describes his needs better. Please note, that, unlike other contests, client here selects not ‘submission’ but ‘Co-Pilot’. So, the information about submission will not be anonymous for the client. Client must be able to see previous history of the Co-Pilot work. For example, if one’s Game Plan was always an underestimation, then client can think that the budget can be exceeded this time as well.

There will be no formal rules for client to fill, but there will be a list of recommended questions to answer during Co-Pilot selection:

* Does the provided documentation meet project timeline and budget?
* Does the content of development strategy document meet application needs?
* Does the development strategy document prove the Game Plan?
* How experienced submitter is?
* Has submitter any current projects?
* Has submitter positive history?

There will be no appeal phase for the screening and Co-Pilot client selection.

* *Co-Pilot Reliability*

Co-Pilot reliability is much more important issue than reliability in other contest tracks. Instead of providing bonuses to reliable members (as in other tracks), unreliable Co-Pilots will be punished. There are following options:

* Co-Pilot contest registrant did not submit Game Plan and Development Strategy. In this case member will be unable to take any Co-Pilot opportunity during next 60 days.
* Co-Pilot discards an opportunity after he was selected (before the actual project start). In this case, member will be unable to take any Co-Pilot opportunity during next 90 days.
* Co-Pilot leaves the project in the middle, with some warning (i.e. he writes letter to PM with information, that he will be unable to complete the project). In this case, member will be unable to take any Co-Pilot opportunity during next 120 days.
* Co-Pilot stops working on the project without any warning. In this case Co-Pilot will be removed from Co-Pilot pool forever.
* *Co-Pilot Payments*

Criteria

Following criteria must be taken into account:

* Quality of proposed Game Plan. The more precise initial Game Plan was the larger must Co-Pilot payment be.
* Responsiveness of work. Co-Pilot must be active in forums and monitor all managed contests.
* The most important factor: the quality of obtained result.
* The positive Co-Pilot history must be accounted. The idea and relative size of this bonus will be similar to reliability bonus in other tracks.
* Project complexity. The larger project is the higher Co-Pilot compensation for one contest must be. This is because for larger project Co-Pilot will need to analyze more information and have the whole project image in mind.
* At the same time, the formula for calculating earnings must not be very complex. Potential Co-Pilots must be able to understand easily, how much money they will earn for the project.

Now several different approaches will be proposed. They’ll be given some codenames to refer them easily. Some approaches are more suitable for large projects, others for small ones.

Current payment (roughly) is equal to 25% of contest prizes. New system will have average earnings be similar in size, but these earnings will vary according to above factors.

Completion Model

This model will be oriented on the produced result mostly. Here’s a description.

* Co-Pilot will obtain fixed fee per contest equal to 20% of contest price (i.e. 400$ for component competition). This is applicable only to contests mentioned in Game Plan. 50% will be paid after contest finish, 50% on successful project completion.
* If any additional contest is required (not mentioned in Game Plan), the payment will be 15% of contest first prize in compare with 20% for normal contest.
* In case of contest repost, larger part of payment will be moved to project completion (i.e. 40% will be paid on contest completion, and 60% on project completion). This must be done because there’s a risk that result of the contest will not have enough quality. The size of Co-Pilot payment will remain the same after contest prize increase (i.e. if 600$ design was reposted for 800$, Co-Pilot will receive 20% of 600$ - design).
* Payment for each bugrace that was planned is equal to 50% of the bugrace prize. It should be large. For each not planned bugrace, Co-Pilot will receive 10%.
* In case of leaving the project, Co-Pilot will not receive any payments planned for future.
* After completion, following bonuses/cuts can be applied to the payment:
* Project size bonus. If project size was large (i.e. number of contests is larger than 35) 10% bonus will be added to initial payment.
* Reliability bonus. If member has 5 or more successful Co-Pilot project in his history, 10% bonus is added. If member has 10 or more successful Co-Pilot project in history, 15% bonus is added.
* In case of some serious troubles, that, on the other hand, did not prevent project from successful completion (for example, Co-Pilot haven’t responded for three days), his payment will be decreased by 2% for each such issue.

Sample:

Experienced Co-Pilot with 7 successful projects in history has created a Game Plan with 16 contests:

* Conceptualization (2000$ )
* Specification (2000$)
* System Architecture (2000$)
* Module Architecture (2000$)
* 5 Components (2000$ each)
* Assembly (2000$)
* Testing Scenarios (300$)

20300 prizes in total.

Also, 5 bugraces were planned.

In reality, one additional design/development was required. 10 Bugraces for 50$ each were run.

Co-Pilot payment for such situation will be:

Initial payment: 20300 \* 20% = 4060$

Payment for additional component: 2000 \* 15% = 150$

Payment for planned bugraces: 250 \* 50% = 125$

Payment for not planned bugraces: 250 \* 10% = 25$

The sum is: 4360$.

Member will have experience bonus 10%: 4360 \* 10% = 436$

Total payment will be: 4796$

Greedy Model

This model is based on the fact that client wants to pay less and Co-Pilot wants to earn more. In this model Co-Pilot will be paid for each contest that was not required to have. This model will require Co-Pilot to be more experienced and professional in both technical and management and look deeper in the details.

In this model payment structure will be defined as follows:

* Game plan will be a ‘worst-case scenario’. Each Game Plan economy will cause bonus payments, each Game Plan exceed will be highly penalized.
* For each contest mentioned in the Game Plan, Co-Pilot will receive 15% of the first prize. 60% of it will be paid after contest finish, 40% - after project completion.
* For saved money (by reducing contest cost or by decreasing contest number) ‘saved’ money will be distributed between client and Co-Pilot. TopCoder fee will remain the same. For example, component production estimated cost is 2000$. It includes design, development and review. Reduce design cost from 600$ to 450$ (and development from 400 to 300) will reduce total cost from 2000$ to 1500$. Co-pilot will receive 250$ and client will pay 250$ less. This money will be paid after project completion.
* For each failed contest Co-Pilot will be penalized. Each reposted contest will not be eligible for ‘economy bonus’. The payment for such contest will be only 10% of the contest prize.
* For each extra contest 10% of the contest prize will be paid.
* In case of leaving the project, Co-Pilot will not receive any payments planned for future.
* After completion, following bonuses/cuts can be applied to the payment:
* Project size bonus. If project size was large (i.e. number of contests is larger than 35) 5% bonus will be added to initial payment.
* Reliability bonus. If member has 5 or more successful Co-Pilot project in his history, 15% bonus is added. If member has 10 or more successful Co-Pilot project in history, 25% bonus is added.
* In case of some serious troubles, that, on the other hand, did not prevent project from successful completion (for example, Co-Pilot haven’t respond for three days), His payment will be decreased by 2% for each such issue.

Co-Pilot with 3 successful projects in history has created a Game Plan with 16 contests:

* Conceptualization (2000$ )
* 2 \* Module Specification (2000$ each)
* System Architecture (2000$)
* 2 \* Module Architecture (2000$)
* 10 Components Designs (2000$ each)
* 2 \* Module Assembly (2000$ each)
* System Assembly (2000$ each)
* Testing Scenarios (400$)

Total cost: 38400$

Also, 10 bugraces were planned.

In reality, 8 components were created and Module Assemblies cost 1600$.

It means, that 4800$ was saved.

All 10 bugraces were used.

Co-Pilot payment will be:

Initial payment for run contests: (38400 - 4800) \* 15% = 5040$

Payment for saved money: 4800 \* 50% = 2400$

Payment for planned bugraces: 500 \* 50% = 250$

The sum is: 7690$.

* *Co-Pilot contest tracking*

Currently it is hard to track contest status for all projects that Co-Pilot is involved in.

There’s no page, where Co-Pilot can see what is happening with his projects and what project needs additional attention. New system will introduce such special page. Short description will follow. Provided mock-ups are just drafts and must be elaborated during specification stage.

The page will show a list of contests. The list can be grouped by contest type or by project. Co-Pilot must be able to switch from one view to another quickly (using button or dropdown box, without going to any settings).

Information for each contest will contain:

* Contest type (design/development/…)
* Contest start date and time
* Contest finish date and time
* Current stage (registration/submission/review/…)
* Link to contest page
* Link to contest forum
* Link to contest online review
* Number of rated/unrated registrants
* Is there any high-rated and reliable registrant (i.e. rating > 1500 && reliability == 100). These numbers must be able to be adjusted for each contest type, because there’re several red designers but no red architects.
* Number of submissions (with submitters handles)
* Links for submission downloads directly from this page
* Forum activity (number of posts in last 24 hours and number of new posts)
* Number of not-answered messages if it is possible. As a first assumption, it could be number of threads where last post’s author is one of registrants.
* Number of reviewers registered.
* Winner/second place
* Prize and DR points for the contest.
* Problems area (see detailed description below)
* Attention area (see detailed description below)

Problems area will contain a list of problems that must be solved. It is hidden if there’s no problem with the contest. Each message in this area will have a link associated that can be used to resolve the issue.

Problems include:

* Screening/Review has been started but there’s no screener/reviewer registered. Link will follow to Review Opportunities page.
* Screening/Review/Aggregation is late. List of reviewers who did not submit their work in time will be shown.
* Final Fixes are late. Handle of submitter will be shown.
* There was no answer for questions for more than 24 hours on forums. Link to the forums will be shown.
* Registration has been closed but no rated member has registered. Link to ‘registrants’ page will be shown.
* There were no submissions for the contest.

Attention is less strong than problem. It can just ‘warn’ about something.

Attentions include:

* Submission stage is closing in 24 hours but there’s no submission. However, if 2 or more members with reliability 100% are registered, the attention will be shown only one hour before submission deadline.
* Review/Screening/Aggregation is closing in 2 hours but there’re left deliverables
* Submission is closing in 24 hours but reviewer has not been registered
* There’re non-answered questions in the forum.

The list of running contests can be quite large, so, all this information must be shown in ‘expanded view’ for the contest. In usual view there will be only one line for each project.

The example of contest list (grouped by project).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Contest Name** | **Contest Type** | **End Date** | **Status & Expand sign** | |
| **Project #1** | | | | |
| Design Contest #1 | Design | 28.12.2010 | | Running > |
| Development Contest #1 | Development | 29.12.2010 | | 2 Problems > |
| Assembly Contest #1 | Assembly | 30.12.2010 | | 3 Attentions > |
| **Project #2** | | | | |
| Design Contest #2 | Design | 31.12.2010 | | 1 Problem > |
| Design Contest #3 | Design | 20.12.2010 | | Completed – Success > |
| Design Contest #4 | Design | 21.12.2010 | | Completed – Fail > |

The example of expanded contest details:

|  |  |  |  |
| --- | --- | --- | --- |
| Concept Contest #1 | Conceptualization |  | 1 Problem ^ |
| Start Time: 10.12.2010 10:00  End time: 12.12.2010 (35 hours left)  Current stage: Review (ends in 12 hours)  Prize for the first place: 1000$  DR Points: 450  [Contest page](http://www.topcoder.com/tc?module=ProjectDetail&pj=30011486)  [Contest forum](http://forums.topcoder.com/?module=Category&categoryID=6799): 10 new messages in last 10 hours, 2 not answered questions  [Online Review](http://software.topcoder.com/review/actions/ViewProjectDetails.do?method=viewProjectDetails&pid=30011486)  Registrants: [3/22](http://www.topcoder.com/tc?module=ViewRegistrants&pj=30011486)  No high-rated and reliable member registered  2 submissions:  [abkqz](http://www.topcoder.com/tc?module=MemberProfile&cr=15795581&tab=concept) [download link]  [iogy](http://www.topcoder.com/tc?module=MemberProfile&cr=22841197) [download link]  1 reviewer registered:  [Ghostar](http://www.topcoder.com/tc?module=MemberProfile&cr=151743) | | | |
| Problems:  The question is not answered in forum: [link to forum] | | | |
| Attention:  Review is closing less than in two hours | | | |

The description of coloring used:

|  |  |
| --- | --- |
| Color | Description |
| Bright Red | Problems section, failed contest |
| Bright Yellow | Attentions section |
| Red | Contest with problems |
| Yellow | Contest with attentions |
| Green | Successfully finished project |

As a suggestion, email notification could be send on any problem occurred.

* *Time tracking*

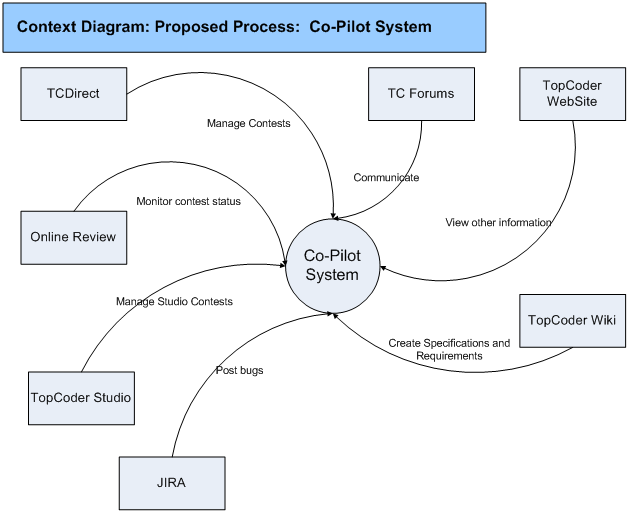
Existing system lacks the ability for easy looking the project timeline. PM uses some special tools (like MS Project), but Co-Pilot does not have access to it. To improve the process, export to portable Calendar format can be used. The tracking must be set up in Cockpit to allow scheduled contest runs. This timelines must be able to be exported to iCal/Outlook/Google Calendar.

* *Game Plan Online*

In contrast with current approach, when Gameplan is submitted into Online Review as an Excel file, online viewing of GamePlan should be created. This functionality will be implemented later. Such addition will help clients to see Gameplan easily.

* **Context Diagram**

The Context Diagram illustrates the modules, business processes, etc that feed or interact with this proposed application.

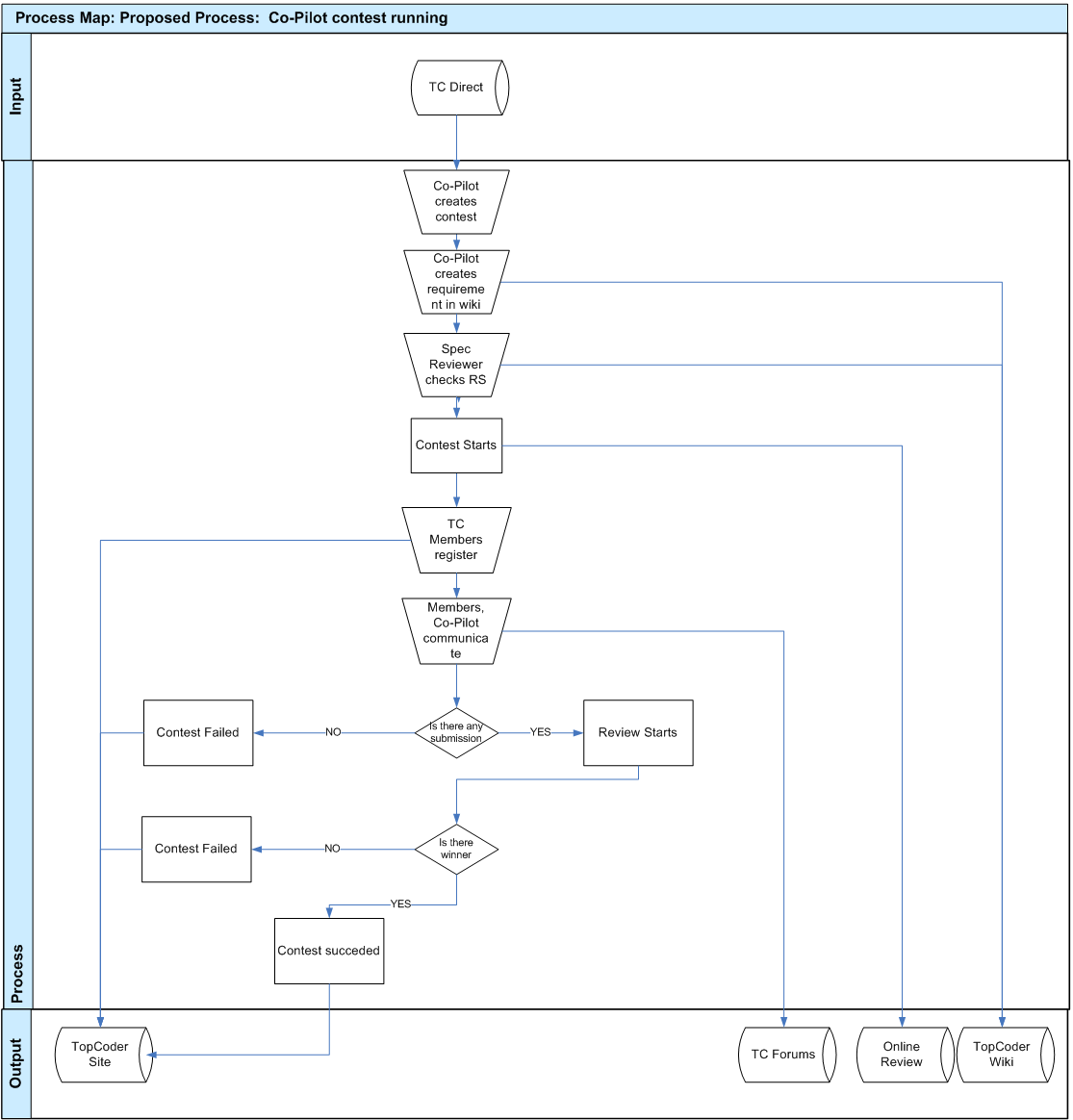
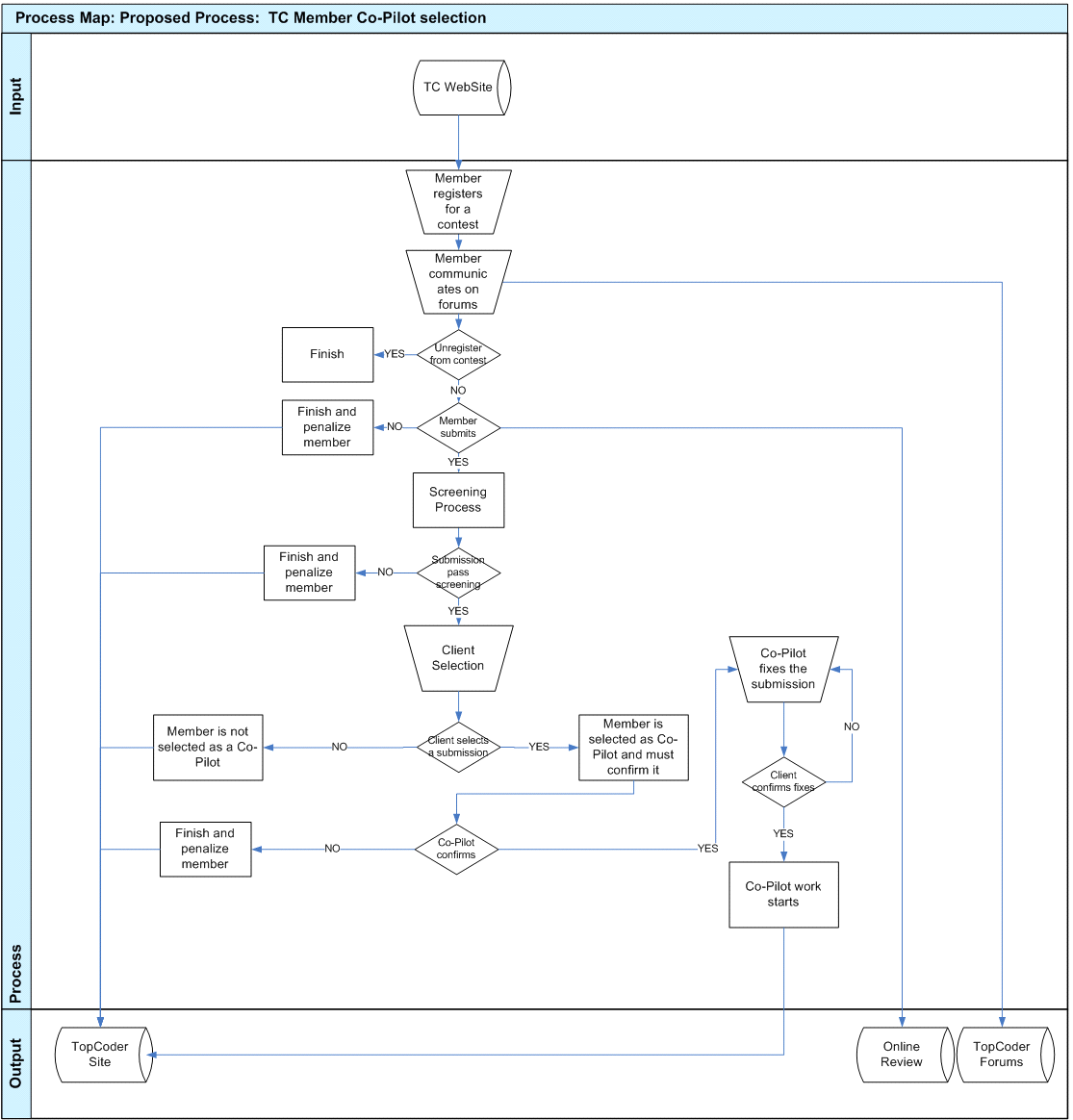
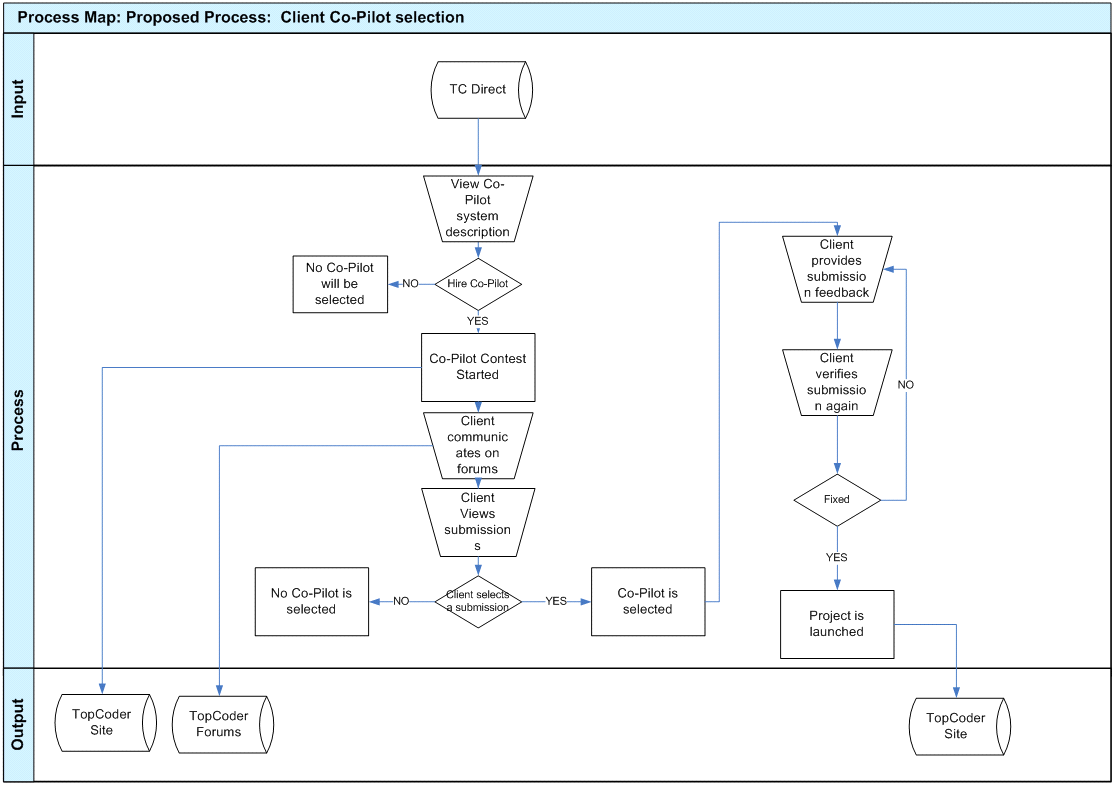


* **High Level Workflow**

The workflow required to complete the primary objectives of the proposed application is described below. The workflow is business-centered, and includes “decision forks” for decisions the business user, or application, must make to achieve the objective. The workflow omits application faults or exceptions. Individual tasks in the workflow are described.

* *Workflow/Process Map Diagram*

These diagrams describes the same process – Co-Pilot selection, but, from different points of view: member and client.



* *Workflow Description*

Client Co-Pilot Selection

* When client creates project he will see description of Co-Pilot opportunities
* Problems: since TopCoder software production model is quite different from classic approaches, it is possible that client will not understand why he needs a Co-Pilot
* Opportunities: some textual examples and graphic could be created for this section. A kind of PowerPoint presentation would be useful as well
* If client desires to use Co-Pilot, the contest will be created.
* Opportunities: The interface for that must be intuitive and straightforward.
* After contest starts, customer must discuss the contest on forums
* Problems: Client can have not much time to answer all questions in time. In this case contest should be extended.
* After submission and screening, a number of submissions will be presented to client. He can select one of them or discard all submissions.
* Problems: Client may select wrong Co-Pilot based on low budget/small timeline. This must be clearly explained to client.
* Client provides feedback to the winning submission and waits for fix
* Problems: some new details can be figured out on this stage
* Client checks final submission. If everything is fine, contest is launched. Otherwise, new fix must be applied.

Member Co-Pilot Contest

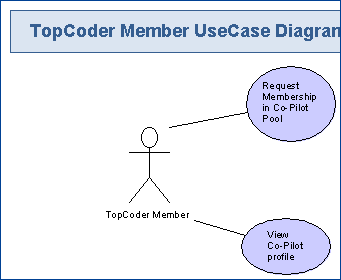
* Member registers for the contest. Only members from Co-Pilot pool are allowed to register.
* Member communicates with client on forums.
* Problems: TopCoder community is more technical than business-analyst. The ‘right’ questions for such contests can be very different from technical-based contests.
* During registration phase member is allowed to unregister
* Problems: member can forget to unregister and will be seriously penalized. It must be clearly stated on registration
* If member is not unregistered he must submit and pass screening. See 3.1.5 section for reliability penalizations.
* Problems: contestant may want to appeal, but appeals are not allowed here.
* If client does not select member’s submission, the process ends.
* Problems: none;
* If client selects member as co-pilot, member must confirm it. If he fails to do it, he will be penalized (see 3.1.5 for details)
* Member must fix all issues in his submission described by client.
* Problems: new details can be found out on this stage
* If client confirms fixes, Co-Pilot work starts. Otherwise, Co-Pilot must fix the submission again.

Co-Pilot Contest Management

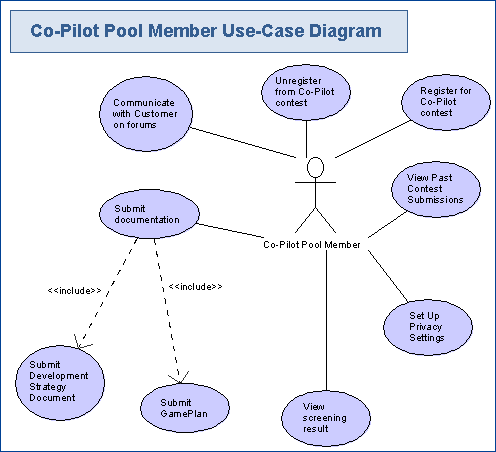
* Co-Pilot creates the contest.
* Co-Pilot writes Requirement Specification.
* Specification Reviewer checks the requirements
* Contest starts at scheduled time.
* TC Members register
* Problems: there can be a few registrants;
* Members and Co-Pilot communicate on forums
* Problems: Due to different time zones communication can be slow;
* If there was no submission, contest is failed.
* Problems: the failure is problem itself. Contest will possibly be reposted.
* If no submission passes review, contest is failed.
* Problems: the failure is problem itself. Contest will possibly be reposted.
* Otherwise, contest succeeded.
* Problems: the quality of submission should be checked by Co-Pilot.
* **Use Case Diagrams**

Use case diagrams are helpful illustrations of how the application’s users interact with the system. Use case “bubbles” correlate to distinct task domains and are useful in determining user roles, module boundaries, shared tasks, etc. The use case diagram are intentionally high level, and exclude activity details such as for example “submit shopping cart.” The diagram below should be used to verify that the required functionality of the application is covered.

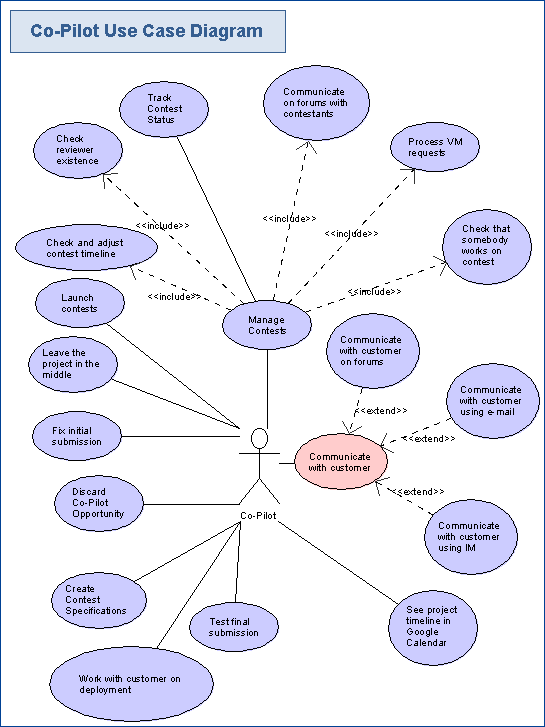
TopCoder Member UseCase Diagram



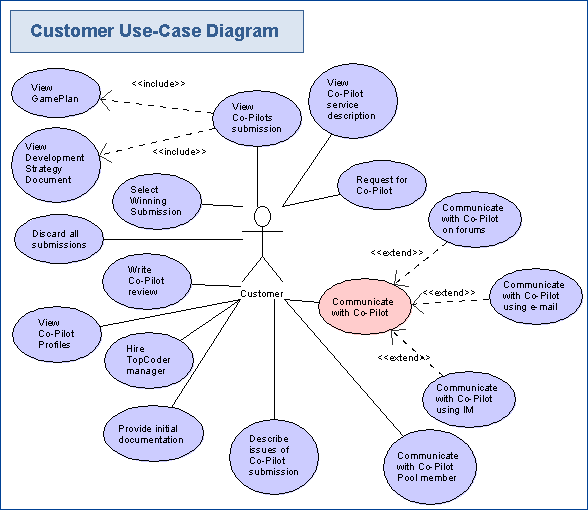
Co-Pilot Pool Member Use-Case Diagram



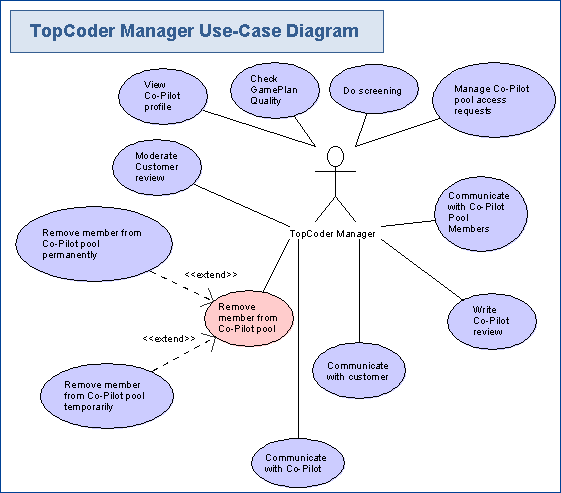
Co-Pilot Use-Case Diagram

 F

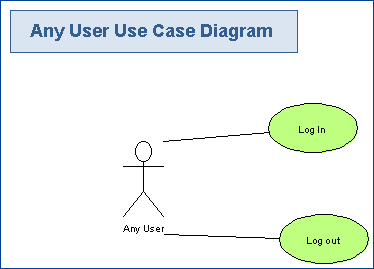
Customer Use-Case Diagram



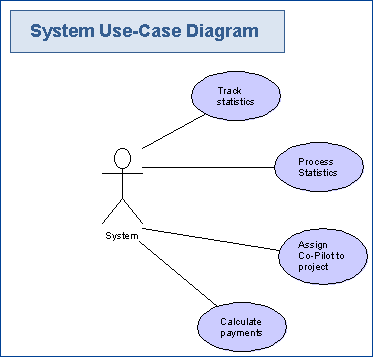
TopCoder Manager Use-Case Diagram



TopCoder Manager Use-Case Diagram



System Use-Case Diagram



* **Business Requirements**

This section identifies, enumerates and explores the business requirements that must be met by the application. Business requirements include capturing the types of users, the basic inputs and outputs, the system’s dependencies, and the tasks the system should accomplish. Some business requirement for a web-based retail site might include:

Users: “The system supports public browsers, registered members, customer service administrators, and system administrators.”

Tasks: “Users have personalized accounts,” “shopping carts expire in a configurable period, and the default is 10 days”, etc.

It is important to confirm that Task Requirements include all the tasks required to meet the business objectives.

* **Users**

All applications have users and most have several users of different types. This section identifies, at a high level, the *types* of users of the system.

* *Customer*

This user is TopCoder client who wishes to hire a Co-Pilot for his project.

* *TopCoder Project Manager*

This user is a TopCoder employee that will be the main responsible person for the project. Co-Pilot will help him. Manager is responsible for checking Co-Pilot work.

* *TopCoder Member*

This is TopCoder member who is not added to Co-Pilot pool.

* *Co-Pilot Pool Member*

This is TopCoder member who is able to register for Co-Pilot positions.

* *Co-Pilot*

This is user who selected for Co-Pilot position.

* **Inputs/Outputs**

This section identifies and describes the inputs to and outputs from the new application. Inputs can include electronic inputs, like RSS and EDI feeds, updates from external databases, etc, as well as human inputs, like “user X keys in results from report Y.” Output can include electronic feeds, printed reports, etc. In this section, all electronic inputs and outputs are captured. Human inputs are omitted from this section.

* *Inputs*
* TopCoder past contest statistics for Co-Pilot profiles
* TopCoder forum posts
* Final project timeline documents
* TopCoder Contest Statuses and Information. It includes registrants, reviewers and other contest-related information.
* Login/logout information
* *Outputs*
* Google Calendar Events
* Email notifications from TopCoder website
* Created product and documentation
* New Contest statistics
* Payment provisioning
* Requirement specifications
* **Dependencies**

Human inputs are also categorized as dependencies. List all dependencies.

* *Client dependency*

System depends on Client actions and provided documentation. Client actions include forum posts, other communication with Co-Pilots, review of Co-Pilot contest submissions and writing Co-Pilot feedback

* *Co-Pilot dependency*

Co-Pilots will submit documentation, provide answers on forums and communicate with Client and PM using all means of connection.

* *PM dependency*

PM will input Co-Pilot feedbacks, screening results, review of Client feedback and other communication.

* *TopCoder wiki*

Wiki will be used for specification creation and review.

* *TopCoder forums*

Forums will be the main communication tool for the system.

* *TopCoder database*

Current database will be used as data source for Co-Pilot statistics fill.

* *TopCoder Direct*

TopCoder direct will be used by client to manage and observe the process and create initial contest.

* *TopCoder Online Review*

Online review will be used for contests submissions.

* *Google Calendar*

Google calendar will be used for timelines export.

* *TopCoder bugtracker (JIRA)*

TopCoder Jira will be used for BugRaces, deployment tasks and as a result of competitor contact (when user contact manager from OR a bug is created automatically).

* *Communication tool*

Communication tool described in 3.1.2 will be used for easy communication.

* *TopCoder website*

It will be used as a base for UI implementation for this module.

* *TopCoder Studio*

Will be used for Studio contest status tracking and management.

* **Task Requirements**

Task Requirements correlate to the use cases, above. There should be at least one task for each use case bubble. Each use case should be briefly described. If there are specific requirements related to the use case, they should also be listed. This section captures high-level, basic functional requirements. Detailed requirements will be defined during the Requirements Specification contests.

TopCoder Member Requirements

* *Request membership in Co-Pilot Pool*

Each member can request membership in Co-Pilot pool. Each request will be processed by TopCoder managers. Following categories are allowed to enter the pool:

* Members who have previous successful Co-Pilot work history;
* Members who have rating > 1500 in architecture, specification or concept;
* Members who have reliability 100% in architecture, specification or concept and 3 or more contests in these tracks;
* On PM discretion, well-known and qualified designer and developers can also be added to Co-Pilot pool
* *View Co-Pilot Profile*

Each member can see Co-Pilot profile. However, the information will not be full. Member will be unable to download previous submissions.

Co-Pilot Pool Member Requirements

* *Communicate with Customer on forums*

When Co-Pilot Pool Member registers for a contest, he must communicate with customer on forums to reveal all details of the project and create a reasonable documentation. This is one of the main differences in compare with previous Co-Pilot process, where direct communication with client was very limited.

* *Register for Co-Pilot contest*

Each Co-Pilot Pool Member can register for the Co-Pilot contest. He must submit a gameplan and pass screening, if he didn’t unregister.

* *Unregister from Co-Pilot contest*

During registration phase, each registrant is able to unregister from the contest. This is very important to unregister if member is not going to submit, because penalizations for it will be serious.

* *Submit documentation*

Co-Pilot contest registrant must submit documentation that will be screened by TopCoder and analyzed by client.

* *Submit Development Strategy document*

Development strategy document is a kind of ‘vision’ for the project. It is a merge of conceptualization and system architecture, greatly reduced in size and details. This document will be a proof for the game plan provided.

* *Submit Game Plan*

Game plan is a document that defines project timeline and cost. Although game plan is a subject to change, it is important to create as precise game plan as possible initially, since this is one of factors for Co-Pilot work quality measurement.

* *View screening result*

After screening, submitter can see screening result. No appeal phase will be available, so documentation must be as clear as possible.

* *Set up privacy settings*

In TopCoder profile there will be a separate setting – hide Co-Pilot earnings. This will be separate from usual Payments number.

* *View Past Contest Submissions*

Each Co-Pilot pool member can see previous submissions on previous Co-Pilot selection contests. This will help to provide valuable submissions.

Co-Pilot Requirements

* *Manage contests*

Co-Pilot’s main task is to help with contest management. It includes different aspects of the TopCoder software development process, so, the experience in different types of contests is encouraged for the Co-Pilot. He must be familiar with all available competition tracks.

* *Check and adjust contest timeline*

The timeline for the competition can be adjusted, if some member asked to extend the deadline. The extension will not affect Co-Pilot payment, but, in case of large number of extensions, overall project timeline will be extended and this will be seen in Co-Pilot profile page. Possibly, it is one of the main things the client will check when selecting co-pilot – how precise he’s able to meet the timeline.

This use-case also corresponds to notifying submitters/reviewers about late reviews/final fixes.

* *Check reviewer existence*

Each contest must have reviewers registered. If reviewer is not registered, broadcast mail to review board members must be sent to ensure that timeline won’t shift because of empty reviewer slot.

* *Fix initial submission*

Client will describe issues found in initial submission. Co-Pilot must fix them in order to start work on project.

* *Track contest status*
* Co-Pilot will be able to see contest statuses for his project on one page. This information is detailed in 3.1.9
* *See project timeline in Google Calendar*

Co-Pilot will be able to see project timeline in Google Calendar on his web-browser or mobile-phone. This information is detailed in 3.1.10

* *Communicate on forums with contestants*

One of the main tasks is to assist contestants. Often, other member is responsible for answering questions (i.e. architect answers designer questions, designer answers developer questions), but, Co-Pilot is the only member who follows project from the beginning to end and he must be able to address any complex and not trivial issue.

* *Process VM requests*

Development is often done on Virtual Machines provided by Amazon. Co-Pilot is responsible for VM assignments to competitors.

* *Check that somebody works on the contest*

If forum is inactive no one is going to submit, Co-Pilot must communicate with registrants and ask about submission perspectives. In case that anyone needs extension/more information, required help must be given.

* *Launch contests*

Co-Pilot will be responsible for launching the contests and gathering all required documentation for them.

* *Leave the project in the middle*

It can happen that Co-Pilot is unable to continue working on project. It is very bad for the project, however, this possibility must be taken into account. The Co-Pilot who leaves the project in the middle will be penalized seriously and lost all his payments delayed for project finish.

* *Discard co-pilot opportunity*

After client selection, Co-Pilot can discard the position. Such behavior is highly discouraged and will be penalized, however, it is better than leaving project in the middle. The penalization will not be as strong as in 4.4.19 case.

* *Work with client on deployment*

The deployment on client environment is Co-Pilot responsibility. He must ensure that software was correctly deployed and use bug-races if any small problems arise. In case of serious errors that prevents projects from being deployed, additional contests could be made, but this will lead to lower Co-Pilot characteristics. In cases when deployment work is not very simple (this happens quite often), special Deployment task can be created by Co-Pilot.

* *Test final submission*

The final distribution must be tested by Co-Pilot. It includes both testing on TopCoder VM and client environment.

* *Communicate with customer*

The key for project success is good communication skills. Co-Pilot must ask client early about each problem/possible choice.

* *Communicate with customer using IM*

To allow faster communication than in e-mail or forums, client and Co-Pilot can exchange their IMs and use them. However, this way is discouraged, because it will be hard to track conversation history.

* *Communicate with customer on forums*

Communication on forums is still important, because allows to include community opinions.

* *Communicate with customer using e-mail*

E-mail conversations are still a most convenient and traditional way to communicate and will definitely be used. However, this way is discouraged, because it will be hard to track conversation history.

Customer requirements

* *View Co-Pilot service description*

Client must be able to get clear description about what is a ‘Co-Pilot’, what are benefits of Co-Pilot and how much will it cost. Some descriptive graphic can be created for this in addition.

* *Request for Co-Pilot*

Client can request for a Co-Pilot and ask for a special copilot selection contest. The client will choose one of submitters as Co-Pilot for his project.

* *Communicate with Co-Pilot*

The key for project success is good communication skills. Client should communicate with Co-Pilot on all project-related questions.

* *Communicate with Co-Pilot on forums*

Communication on forums is still important, because allows to include community opinions.

* *Communicate with Co-Pilot using e-mail*

E-mail conversations are still a most convenient and traditional way to communicate and will definitely be used.

* *Communicate with Co-Pilot using IM*

To allow faster communication than in e-mail or forums, client and Co-Pilot can exchange their IMs and use them.

* *Communicate with Co-Pilot Pool Members*

On the Co-Pilot selection contest, client must answer questions on forums to allow better project understanding.

* *View Co-Pilot profiles*

Client should view submitters’ profiles to learn Co-Pilots project history and find out who fits the position best.

* *View Co-Pilots submissions*

Client will view the submissions which passed screening and determine their fit to the project needs.

* *View Game Plan*

Game plan is a document that defines project timeline and budget. On this early stage this document can be not ideally, but it will give a rough estimation. The quality of estimation will be reflected in Co-Pilot’s profile, so customer can determine the usual correctness of one’s submission.

* *View Development strategy document*

This document is a proof for a provided gameplan. It must contain enough technical and business details to allow clear feeling of right way for project implementation.

* *Describe issues of Co-Pilot submission*

Client will describe issues that are required to be fixed in the winning submission (if any). He must do it using TC Direct and/or Forums for communication.

* *Write Co-Pilot review*

After project finish, client can write Co-Pilot review. It will be moderated by TopCoder PMs and shown on Co-Pilot profile page. Reading these reviews can help customer to select Co-Pilot for their needs.

* *Select Winning Submission*

Selection of winning submission must be done not only by quality of that submission, but the history and profile of Co-Pilot. There will be no objective criteria for that selection, client will choose the Co-Pilot.

* *Discard all submissions*

If client likes no submission/submitter than no Co-Pilot will be selected. TopCoder will provide hired Project Manager for that project possibly.

* *Hire TopCoder Manager*

TopCoder can provide PM in addition to Co-Pilot. This will make process more expensive for the client, but the process will be more responsible and reliable.

* *Provide initial documentation*

Client will have to provide initial documentation in questionnaire form defined in 3.1.1

TopCoder manager requirements

* *View Co-Pilot profile*

TopCoder Manager will be able to view Co-Pilot profiles as any other members. In addition, they’ll be able to see hidden fields (as Co-Pilot payments).

* *Check GamePlan Quality*

Checking initial GamePlan and its comparison with real timeline will be project manager responsibility. He will determine that information and will Co-Pilot profile with corresponding values after project finish.

* *Do screening*

Initially no community review board will be created for screening Co-Pilot contests. TopCoder staff will do this work. In future, screening will also be community driven.

* *Manage Co-Pilot pool access request*

Managers will be responsible for populating Co-Pilot pool with new members. Members can be added using common criteria defined above, or, at Manager discretion (he knows that this member is qualified well).

* *Communicate with Co-Pilot Pool Members*

News and Co-Pilot opportunities can be mailed to Pool Members by Project Manager.

* *Communicate with Customer*

Manager must communicate with customer and resolve all problems raised between Co-Pilot and client.

* *Communicate with Co-Pilot*

Project manager can communicate with Co-Pilot. He should notify him about some not answered questions or other problems in cases when client and Co-Pilot didn’t manage to solve the problem by themselves.

* *Remove member from Co-Pilot pool*

In cases of low-quality work or reliability issues member can be removed from Co-Pilot pool.

* *Remove member from Co-Pilot pool temporarily*

In case of reliability issues, member of Co-Pilot pool can be suspended for some amount of time (1,2,3,4 months). After that time member account must become active automatically.

* *Remove member from Co-Pilot pool permanently*

In case of serious issues (leaving the project without any notification), member can be removed from Co-Pilot pool forever without restore rights.

* *Remove member from Co-Pilot pool permanently*

PM can write review of Co-Pilot work. It will be also seen at Co-Pilot profile.

System requirements

* *Track statistics*

All contest-related statistics must be tracked. It includes all contests, all reposts, all successes for the project.

* *Process statistics*

Gathered statistics will be process to present in human readable way in Co-Pilot profile page

* *Assign Co-Pilot to project*

Co-Pilot must be ‘assigned’ to the project in the backend in order to allow him management actions.

* *Calculate payments*

Co-Pilot payments must be calculated according to selected payment model.

* **Security Requirements**

This section documents, at a high level, the basic security requirements of the application. For example, does the system require the user to log in? Should the user’s identity be authenticated on just this system, or against a central authority? Are there any special or unusual security requirements, like fingerprint scanning?

* *Co-Pilot Privacy*

Co-Pilot must be able to hide his earnings.

* *Page Authorization*

Only PM and Co-Pilot of the project must be able to see Project Status pages as it contains links to submission downloads.

* *Logins*

Existing TopCoder login process will be re-used. Co-Pilot will not be required to login additionally.

* *Client*

Client will be logged into TopCoder Direct. It should be allowed to access forums without additional login.

* *Co-Pilot rights*

Co-Pilot management rights must be restricted to the project he runs. He must not be able to manage contest that belongs to other project.

* **Performance Requirements**

Performance Requirements for the application are defined at a high level below. If there are specific requirements for specific features to perform at a quantifiable level, they too are listed below. These requirements will be developed in more detail when the Requirements Specification is written, in a later phase.

* *Web Pages will Load in ~2 seconds or less after initial hit*
* *The system will achieve 99.999% uptime*
* *The system will be available 24x7*
* *For email-based messaging tool ~60 seconds delay is allowed*
* *System will be used by several hundreds of users*
* **Data Migration**

Data Migration describes the data that needs to be moved from an older or external system to the new system, in order for it to operate at launch. Migration also includes data that must be transferred from the new system to another external system. Any data migration requirements are listed below.

Though there’s no data ‘migration’ in common sense, existing data that contains contest statistics must be included in Co-Pilot profile. It includes number of run contests, their results, timeline changes.

Some of the data will be filled manually (like comparison between initial gameplan and real one).