<< RETURN

Accurate Prediction and other Organizational Myths

By Starr Long

Bio:

Starr Long has been in the business of making PC games for over ten years. Alongside Richard Garriott, he was the original Project Director for the commercially successful Ultima Online. Starr worked his way up through the ranks of Origin Systems, Inc. starting in Quality Assurance working on Wing Commander, Ultima, and many other titles for Origin. Most recently Starr was the Producer for Ultima Online 2. Starr is currently working with Richard Garriott on an online game named Tabula Rasa for the Korean online game giant NCsoft, creators of the world's largest online game: Lineage.

Presentation Title:

Accurate Prediction and other Organizational Myths

Presentation Format:

Lecture 60 minutes (or maybe 120 minutes?)

Audience Level:

Intermediate

Talk Type:

Educational

Presentation Abstract:

The speaker will describe strategies for organizing your project so that you can better allocate schedule, resources, and budgets. The speaker will attempt to debunk standard organizational myths like accurate schedule prediction and massive pre-production. The speaker will describe these topics from over 10 years of personal experience as both a QA tester and a producer. The speaker will emphasize the need for organization and discipline in a maturing industry

Intended Audience and Prerequisites:

Attendees that will most benefit from this lecture will be managers who are directing and/or producing mid-size to large scale games.

What is the idea takeaway from this presentation?

Attendees should come away from the lecture with an understanding of how to successfully manage a game from beginning to end. Attendees will learn how to keep things simple and organized in order to achieve their goals. Attendees will know what mistakes to avoid by learning which standard organizational practices are inappropriate for the interactive industry. Finally they should come away with a set of strategies to help them make games, on time and on budget.

PRESENTATION SYLLABUS:

Minimize Pre-Production!

Contrary to standard organizational methods

"No plan survives contact with the enemy"

- Giant, extremely detailed, Design Documents done during preproduction are a waste of time
    - Example: On UO2 we spent almost 3 months building a massive 150 page design doc. We ended up completely changing at least 75% of the design, making almost all of that work useless.
- Only do enough pre-production on the game design to build an overall schedule
- Only do detailed designs in conjunction with the programming team as they are implementing a given system

Step 1: Create a concise basic feature list

- List of features with a short phrase describing it
    - Example: Basic AI: attack, defend, retreat, flock
- Should also include things your game explicitly WILL NOT do
    - Example: No arbitrary item placement

Step 2: Create short descriptions of each feature (max one page)

Step 3: Build out engineering schedule

- Based on the feature list and one pagers
- Include "maintenance" time.
    - 50% average over allotted time for unforeseen issues, bug fixing, etc.
    - Maintenance time is the per feature/task cushion. On UO & UO2 we did not include this in our schedule and we were almost always about 50% over schedule.

Step 4: Build Design and Art schedules based on engineering schedule

- Make any changes to Tech schedule based on feedback from Art and Design
    - Example: Art requires Lighting Model to happen before almost all other client tasks so all the art won't have to be redone. On TR we failed to do this requiring massive rework from art.

Create milestones and deliverables that have clear overall goals

- Use meaningful milestone names vs. old definitions of Alpha, Beta, etc.
    - Example: Milestone 2: Walk & Talk: Characters will be able to walk around a game map & talk to other players
- What is the game play like at the end of the milestone?
- Example: At the end of milestone 3 the player will be able to create a character and equip weapons.

Accurate prediction is a myth!

Budget time per feature, don't allocate time based on design

- EXAMPLE: Budget 6 weeks for Character Inventory, any features that fall outside that six weeks are cut/postponed

Prioritize sub-features / sub-systems within each feature/system

- Minimum required for ship, wish list for ship, etc.
- Use this prioritization to determine which features get completed within budgeted time

Only do detailed scheduling and milestone descriptions for a given milestone during the preceding milestone

- Needs & tasks will change as the product progresses so fleshing out details too early just creates rework.
- Constantly reevaluate your schedule
    - Estimates are valuable for guiding the larger motions of the group, regular analysis of *actual* costs contribute greatly to more accurate prediction in each future phase
    - On TR the Art Director regularly reviews the actual costs of each art asset after it is complete

Discipline!

Keep to a reasonable Team size:

- More than 25 on a team is very risky
- Large teams have trouble communicating and staying in synch
- With larger teams Managers (leads, producer, etc.) spend too much time managing people vs. managing the project
- Start small and bring on team only as needed
    - On TR & UO2 we ended up trying to occupy large teams with preproduction tasks until we had tech to support them. Preproduction tasks that all got thrown away since they were based on incomplete tech information.

Don't expect Managers to contribute content

- This is a slowly dying myth in our industry
- Don't try to base your schedule on content from managers
- Tech director will rarely write code, Art Director won't be painting textures
- Managers will be scheduling, developing technology, directing the team
- Leaders lead, production resources produce
    - On UO my original lead programmer spent his time coding which put managing the programming team on the shoulders of the project director (me) who was not a programmer so therefore had a hard time accurately scheduling the programmers.

Throwing more bodies at a problem rarely solves it.

- On UO & UO2 whenever we fell significantly fell behind schedule we were given additional resources.
    - By the time UO2 was cancelled our team was near 75. Our development director could not get through schedule meetings with the entire team in a single week.
- Balance regular full time resources with contract/temp resources to better match production spikes
    - On UO2 rather than do all our level of detail art ourselves we contracted out with an independent art house (Ballistic Pixel Lab) therefore preventing even more team bloat.

Core hours

- I know everyone will be available for discussions and meetings at a certain time each day
- I suggest 9 AM with 8 hours of working time (not including lunch, breaks, etc.)
    - This gives two blocks of time for real work to be done.

- Time for the industry to grow up
- Contrary to popular belief getting to work on time AND in the morning does NOT prevent creativity
    - See the movie industry for example
- Now that we are getting older more and more of us have families and would like to see them in the evenings.
- What I have seen on teams with no core hours (UO) or core hours that don't start until 10 (UO2) is the following behavior:
    - Stroll in around 10 or 10:30 on average
    - Browse email until 11
    - Lunch at 11:30
    - Work actually starts about 1, might as well have just come in at 1
- The actual times are irrelevant, consistency is the key. i.e. I know everyone will be available for a meeting at 10 AM every day.

NO CRUNCH

- Extended mandatory overtime NEVER makes a better game
- I think it actually slowed us down on UO & UO2 by causing people to make more mistakes
- Instead on TR we are doing limited overtime (2-3 weeks max) towards specific goals like demos, milestones, etc.

Documentation, Code Comments, etc.

- The time of the hacker is over.
- Any programmer could take over any other programmer's work just by looking at documentation and comments
- On UO we had no such standards and new programmers to the team spent months trying to figure uncommented/undocumented code out and then usually just rewriting it completely because that was easier. Of course the rewrites were usually as unstable as the original code leading to a general stagnation in forward progress.
- Currently there are no members of the original UO team on the product
- 2.5 years into live production on UO the first technical document was created! Bad!

Tools (Editors, exporters, etc.)

- Allocate at least 1-2 full time experienced resources just to tools
- On most products I have worked on this was always lower priority than getting game code working. This is a HUGE mistake

Art Pipeline Structuring extremely important very early

- Definition: Getting art into the game
- Find the right tools (try very hard not to write them yourself)
- Make sure those tools work well as a long-term, extendable solution
- Then DO NOT CHANGE IT
- Resource Management Tools are critical part of the pipeline:
    - Alien Brain is a great example of a 3rd party tool that does this
    - Stupidly on TR we wrote our own: PAGman

Features will be cut

- You will need to do it, get ready
- On UO and UO2 we refused to cut features and so we did not have time to polish the product (both balancing and stability)

- Blizzard is one of the best examples of keeping scope of project small and polishing core game experience

Maintain a constant high level of communication:

- The entire team should always be aware of the current status of the project
- Regular reports (daily, weekly, monthly, etc.)
    - Helps to know who is doing what when, better synching of dependencies
    - On TR the entire team does a daily report that goes to entire team
    - On TR the Producer (me) does a weekly summary of major accomplishments & issues for the entire company
- Regular meetings
    - Meetings should be as short as possible
        - On UO2 we had the dept. leads give updates about the status of every single team member which dragged the meeting out too long and forced team to hear data not necessarily relevant to them
    - Meetings should always have stated goal, an agenda, and notes/action items should be taken
        - For TR manager's meetings the Producer (me) always has an agenda and list of action items sent to the managers 1 day prior to meeting
        - On TR the team and individual departments meet on a weekly or bi-weekly basis
        - On TR All managers have bi-weekly 1 on 1s with their reports
        - On TR Producer (me) has 1 on 1s with entire team including QA after each milestone
- Internal website with links to current documentation
    - On TR we use WIKI for the site and it is tied to our asset management tool
- Get out from behind the desk
    - On UO, UO2, and even the beginning of TR the managers spent too much time on email, schedules, and budgets

Play Your Game!

Stable, Fast, & Fun: In that order

- On UO we spent so much time putting in features that the game was slow and unstable forever

Create actual game environment as early as possible

- Fixing bugs always higher priority than new features.
    - On UO & UO2 we let bugs linger while new features went in
    - On TR everyone must clear out their buglist on a weekly basis
- Always have a working version
    - On UO & UO2 we would go weeks and even months without a working version
    - On TR if we go more than 1 week without a new version the entire team is put on making this happen
- Daily Builds
    - Again on UO & UO2 the builds were a manual process that more often than not failed
    - On TR we automated the process in the first 6 months

Weekly Play sessions as soon as possible

- Make sure team provides feedback for these play sessions
- On UO & UO2 we never played the game together
- On TR we play every week, the team provides feedback and we track actions against that feedback

Structure!

Establish a clear hierarchy from the beginning.

- Make sure everyone knows who to go to for decisions

Organize by department with leaders of each (art, programming, design)

- (INSERT ORG CHARTS HERE)

Strike Teams

- Once basic structure of game is complete move to strike teams vs. departmental model:
    - Examples: UI, Combat, Game flow, etc.
- Retain departmental managers for resource allocation, employee reviews, etc.
- Strike teams are temporary
    - Reorganize from milestone to milestone based on needs
- Goal oriented
    - Each strike should have weekly demonstrable goals with one large goal for end of the milestone
- Cross discipline
    - At least one member from each department (artist, programmer, designer)
    - Improves communication that is traditionally hard across disciplines
    - Strikes avoid slogging through process, they are nimble and dynamic, they promote accountability which usually equals results.

QA & Support: Test Early, Test Often!

Involve QA & Support from beginning

- On UO we didn't start testing until right before our first public test
- On UO even then we ignored crippling bugs that QA found
- On TR we have had QA test every single milestone from the beginning of project

Have QA test every milestone deliverable, even if you are developing internally
Require sign off for all deliverables
Make details like code comments and documentation required for deliverable sign-off
Give QA promotion control for builds

Conclusions:

Minimize pre-production
Budget time vs. attempting to accurately predict
Establish and maintain a disciplined environment
Play your game early and often
Establish and maintain a clear yet flexible team structure
Test, test, test

Accurate Prediction and other Organizational Myths

by

Starr Long

Minimize Pre-Production!

- Contrary to standard organizational methods
- “No plan survives contact with the enemy”
  - Giant, extremely detailed, Design Documents done during preproduction are a waste of time
  - Only do enough pre-production on the game design to build an overall schedule
  - Only do detailed designs in conjunction with the programming team as they are implementing a given system

Minimize Pre-Production!

- Step 1: Create a concise basic feature list
  - List of features with a short phrase describing it
    - Example: Basic AI: attack, defend, retreat, flock
  - Should also include things your game explicitly WILL NOT do
    - Example: No arbitrary item placement
- Step 2: Create short descriptions of each feature (max one page)

Minimize Pre-Production!

- Step 3: Build out engineering schedule
  - Based on the feature list and one pagers
  - Include “maintenance” time.
    - 50% average over allotted time for unforeseen issues, bug fixing, etc.
    - Maintenance time is the per feature/task cushion.
- Step 4: Build Design and Art schedules based on engineering schedule
  - Make any changes to Tech schedule based on feedback from Art and Design

Minimize Pre-Production!

- Create milestones and deliverables that have clear overall goals
  - Use meaningful milestone names vs. old definitions of Alpha, Beta, etc.
    - Example: Milestone 2: Walk & Talk: Characters will be able to walk around a game map & talk to other players
  - What is the game play like at the end of the milestone?
    - Example: At the end of milestone 3 the player will be able to create a character and equip weapons.

Accurate prediction is a myth!

- Budget time per feature, don't allocate time based on design
  - EXAMPLE: Budget 6 weeks for Character Inventory, any features that fall outside that six weeks are cut/postponed
- Prioritize sub-features / sub-systems within each feature / system
  - Minimum required for ship, wish list for ship, etc.
  - Use this prioritization to determine which features get completed within budgeted time

Accurate prediction is a myth!

- Only do detailed scheduling and milestone descriptions for a given milestone during the preceding milestone
  - Needs & tasks will change as the product progresses so fleshing out details too early just creates rework.
  - Constantly reevaluate your schedule
    - Estimates are valuable for guiding the larger motions of the group, regular analysis of *actual* costs contribute greatly to more accurate prediction in each future phase
    - On TR the Art Director regularly reviews the actual costs of each art asset after it is complete

Discipline!

- Keep to a reasonable Team size:
  - More than 25-30 on a team is very risky
  - Large teams have trouble communicating and staying in synch
  - With larger teams Managers spend too much time managing people vs. managing the project
  - Start small and bring on team only as needed
- Throwing more bodies at a problem rarely solves it.
  - Balance regular full time with contract/temp resources to better match production spikes

Discipline!

- Don’t expect Managers to contribute content
  - This is a slowly dying myth in our industry
  - Don’t try to base your schedule on content from managers
  - Tech director will rarely write code, Art Director won’t be painting textures
  - Managers will be scheduling, developing technology, directing the team
  - Leaders lead, production resources produce

Discipline!

- Core hours
  - I know everyone will be available for discussions & meetings at a certain time each day
  - I suggest 9 AM with 8 hours of working time
    - This gives two blocks of time for real work to be done.
  - Time for the industry to grow up
  - Contrary to popular belief getting to work on time AND in the morning does NOT prevent creativity
  - Now that we are getting older more and more of us have families and would like to see them in the evenings.
  - The actual times are irrelevant, consistency is the key.

Discipline!

- NO CRUNCH
  - Extended mandatory overtime NEVER makes a better game
  - On TR we are doing limited overtime (2-3 weeks max) towards specific goals like demos, milestones, etc.
- Tools (Editors, exporters, etc.)
  - Allocate at least 1-2 full time experienced resources just to tools
  - On most products I have worked on this was always lower priority than getting game code working. This is a HUGE mistake

Discipline!

- Art Pipeline Structuring extremely important very early
  - Definition: Getting art into the game
  - Find the right tools (try very hard not to write them yourself)
  - Make sure those tools work well as a long-term, extendable solution
  - Then DO NOT CHANGE IT
  - Resource Management Tools are critical part of the pipeline

Discipline!

- Maintain constant high level of communication:
  - The entire team should always be aware of the current status of the project
  - Regular reports (daily, weekly, monthly, etc.)
  - Regular meetings
    - Meetings should be as short as possible
    - Meetings should always have stated goal, an agenda, and notes/action items should be taken
  - Internal website with links to current documentation
  - Get out from behind the desk

Discipline!

- Documentation, Code Comments, etc.
  - The time of the hacker is over.
  - Any programmer could take over any other programmer's work just by looking at documentation and comments
- Features will be cut
  - You will need to do it, get ready

Play Your Game!

- Stable, Fast, & Fun: In that order
- Weekly Play sessions as soon as possible
  - Make sure team provides feedback for these play sessions & you track that feedback
- Create actual game environment as early as possible
  - Fixing bugs always higher priority than new features.
  - Always have a working version
  - Automated Daily Builds

Structure!

- Establish a clear hierarchy from the beginning.
  - Make sure everyone knows who to go to for decisions
- Organize by department with leaders of each (art, programming, design)

Structure!

- Strike Teams
  - Once basic structure of game is complete move to strike teams
    - Retain dept. managers for resource allocation, etc.
  - Strike teams are temporary
    - Reorganize based on needs
  - Goal oriented
    - Weekly demonstrable goals, one large goal
  - Cross discipline
    - At least one member from each department
    - Strikes avoid slogging through process, they are nimble and dynamic, they promote accountability which usually equals results.

QA & Support: Test Early, Test Often!

- Involve QA & Support from beginning
- Have QA test every milestone deliverable, even if you are developing internally
- Require sign off for all deliverables
- Make details like code comments and documentation required for deliverable sign-off
- Have QA test each daily build
- Give QA promotion control for builds

Conclusions

- Minimize pre-production
- Budget time vs. attempting to accurately predict
- Establish and maintain a disciplined environment
- Play your game early and often
- Establish and maintain a clear yet flexible team structure
- Test, test, test