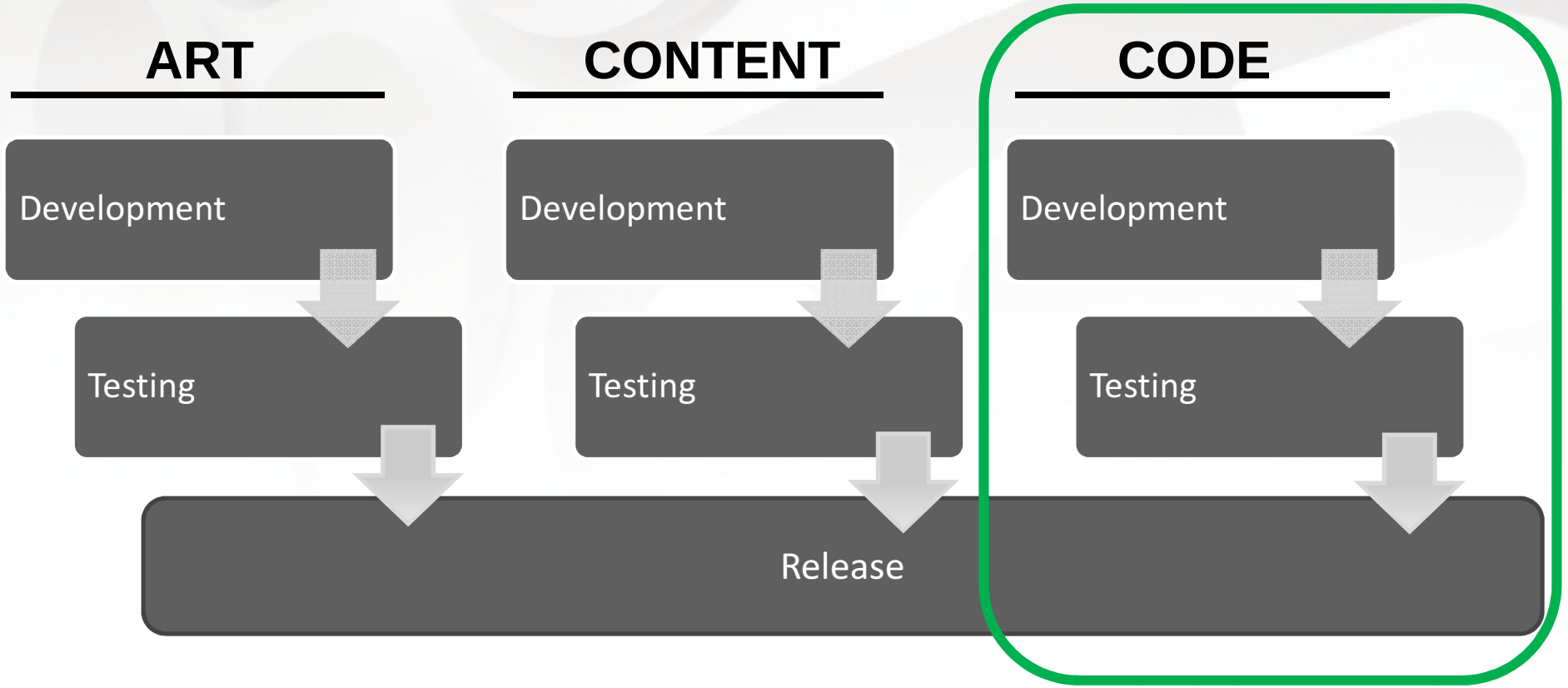
MMO Code and Complexity

Managing EVE's Expanding Universe

Berglind Rós Guðmundsdóttir

EVE Online

EVE Online


ART
CONTENT
CODE
Development
Development
Development
Testing
Testing
Testing
Release

EVE Online

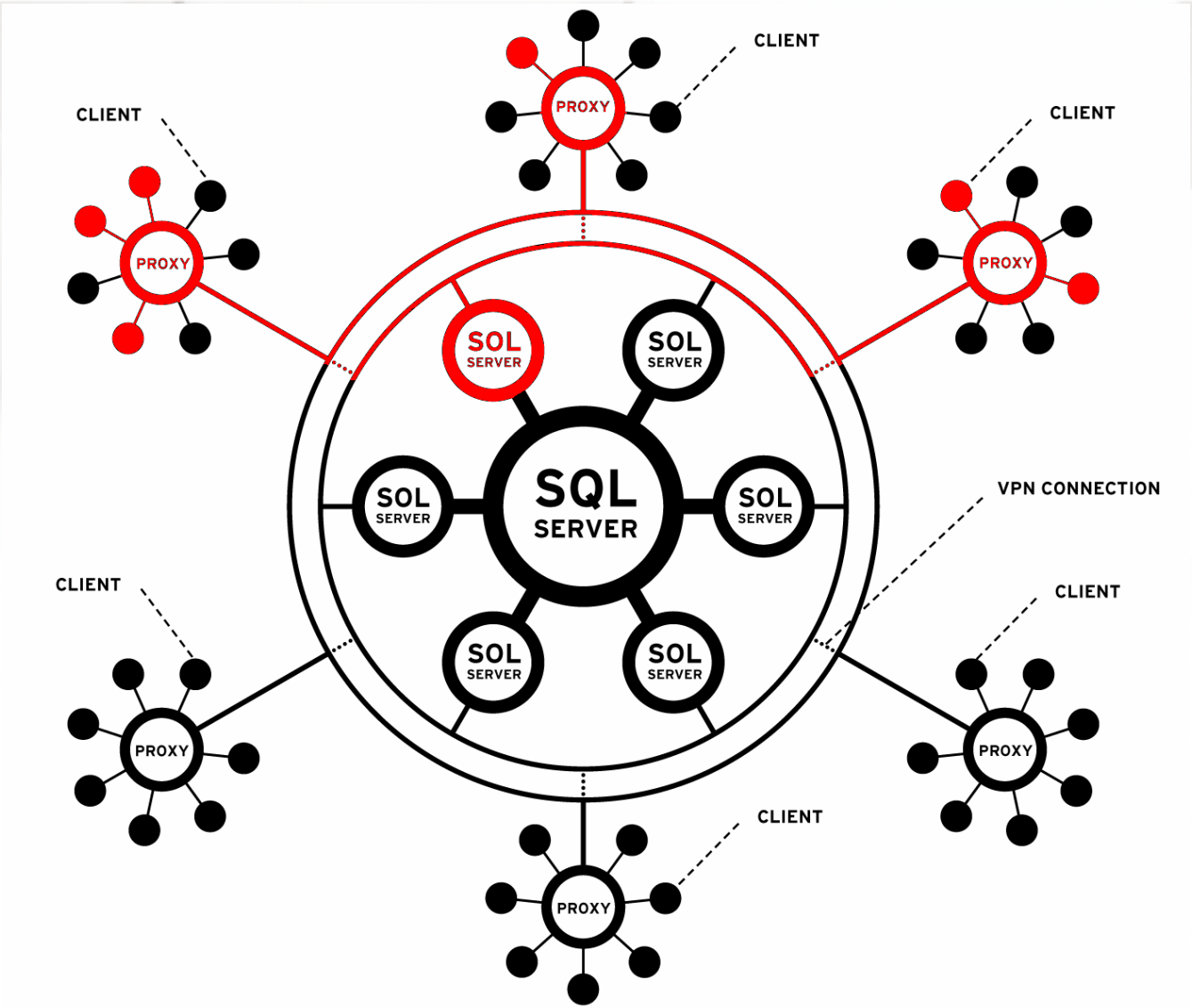
- 330,000 subscribers
- 56,817 concurrent users (Jan 24th 2010)
  - single shard (everyone in the same virtual world)
- Up to 1400 users in most popular solar system
- 1000+ users in fleet fights

- 1,100,000 lines of code
- 4,000 database objects
- 200 server nodes
- 1 database

SQL Server 2008

- Mostly stored procedures (T-SQL)
- Code developed by all, reviewed by DB specialists
- One database
  - Largest table 1,962,000,000 rows
  - 200 million requests/day
  - 2500 transactions/second

Cluster topology



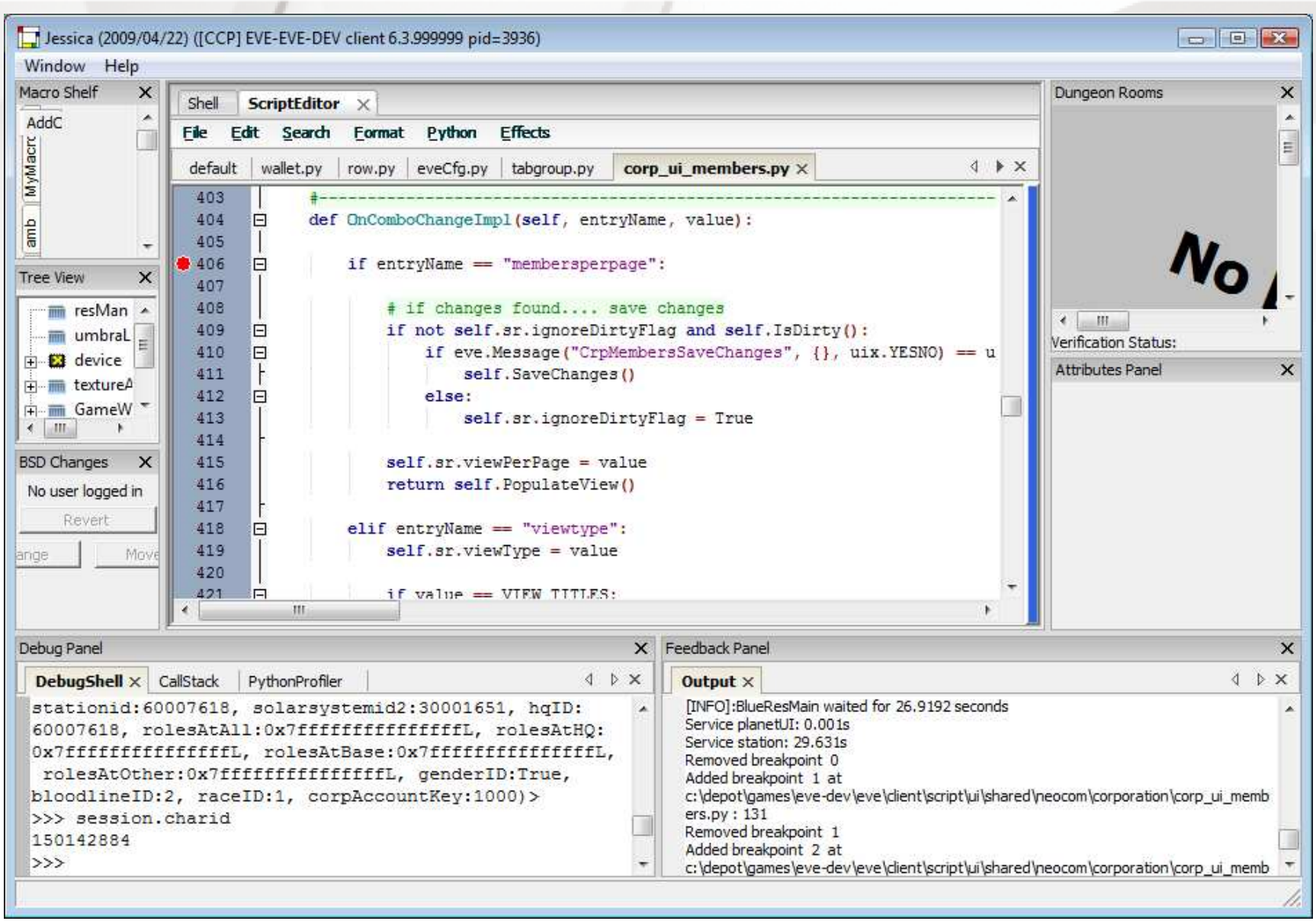
C++

- Performance critical modules
  - Graphics engine
  - Sound engine
  - Solar system physics simulation
  - …
- Longer development time
- Heavier development and build environment

Stackless Python

- Rapid development
- Easily learned
- Flexible
- Light-weight environment
  - TextPad or other text editor of choice

Python development tool

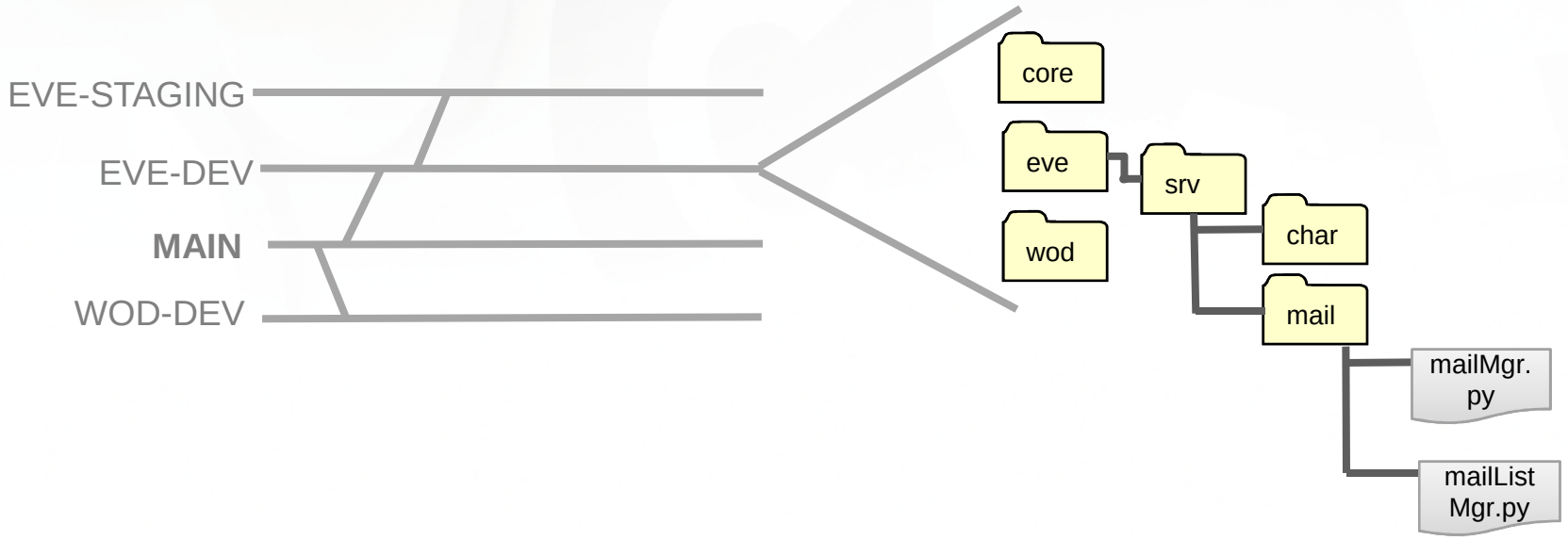


Stackless Python

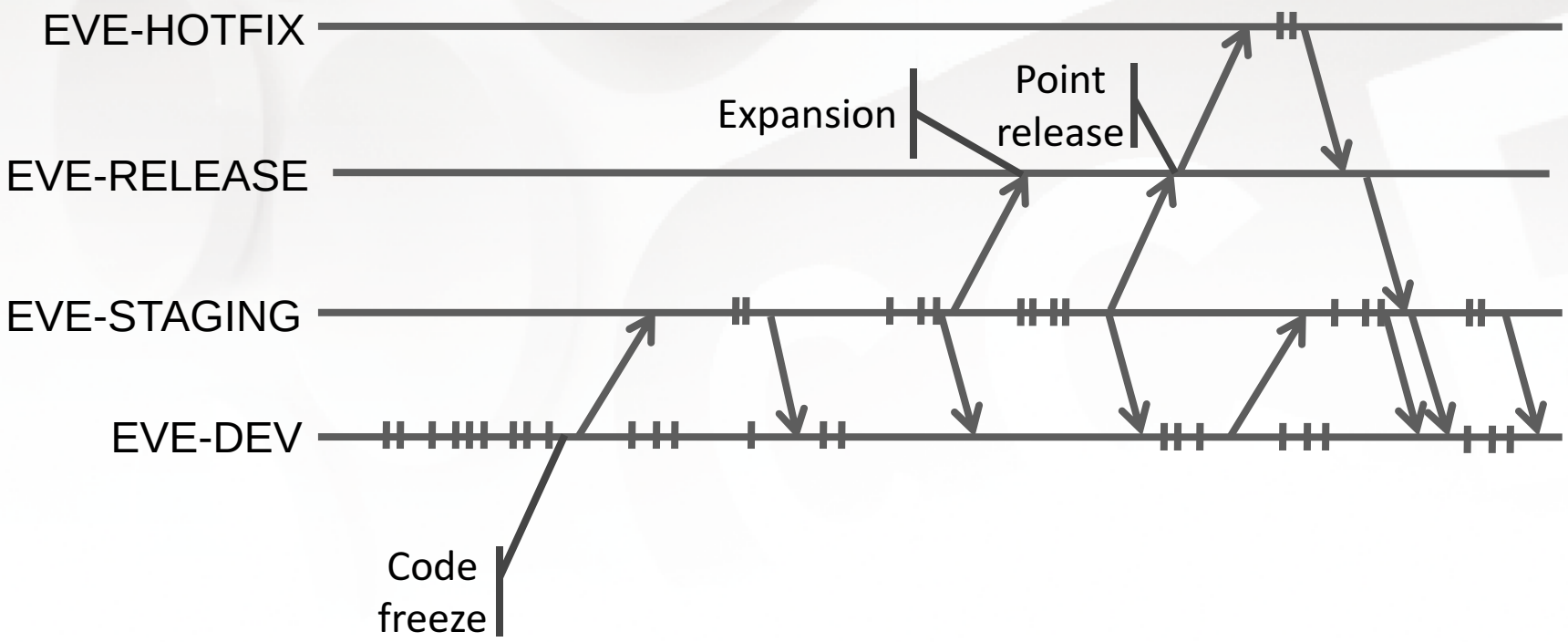
- Not as efficient as a compiled language
- Dependencies may be hard to track
- Compiles ≠ correct
- An error may not surface easily
- Rarely crashes to your desktop!
- Reviews and testing throughout the development cycle

Shared environment

- One single development code base
- One Perforce depot
- Shared core technologies
- Logical separation into modules



Branching using staging streams model



Branched database updates

- Content
  - Agents
  - Missions
  - Spaceship blueprints
  - …
- Inhouse DB implementation of the staging streams model
- Content developers can continue working after content freeze

Non-branched database updates

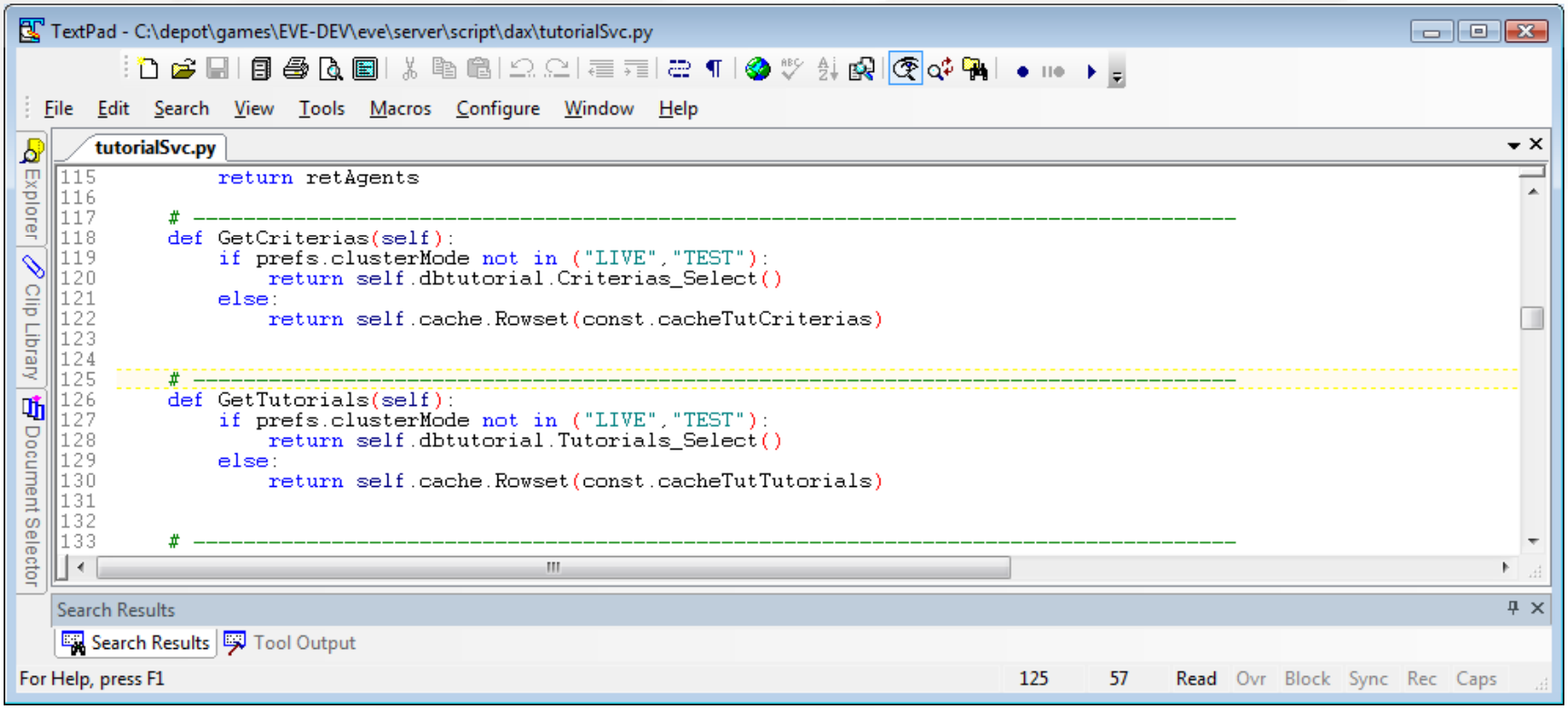
- Static data (solar systems, character races...)
- DB code (procedures, tables, views)
- *Must be*
  - ***backwards compatible***
  - ***released at the correct time***
  - ***deployed in the correct order***
- Sequentially numbered updates
- Linked to code changes with markers
- List of database updates is generated from markers
- As automated as possible

Building

- < 1 minute to build a server
- < 10 minutes to build a client
- < 30 minutes to rebuild all binaries
  - Relevant binaries are automatically rebuilt when C++ code changes
  - Highly optimized build process using Visual Studio
  - Caretaking of the code with regards to build time
- < 4 hours to build all installers and patchers
  - 2 full installers (Windows & Mac)
  - 40 patchers (from 20 versions, Windows & Mac)

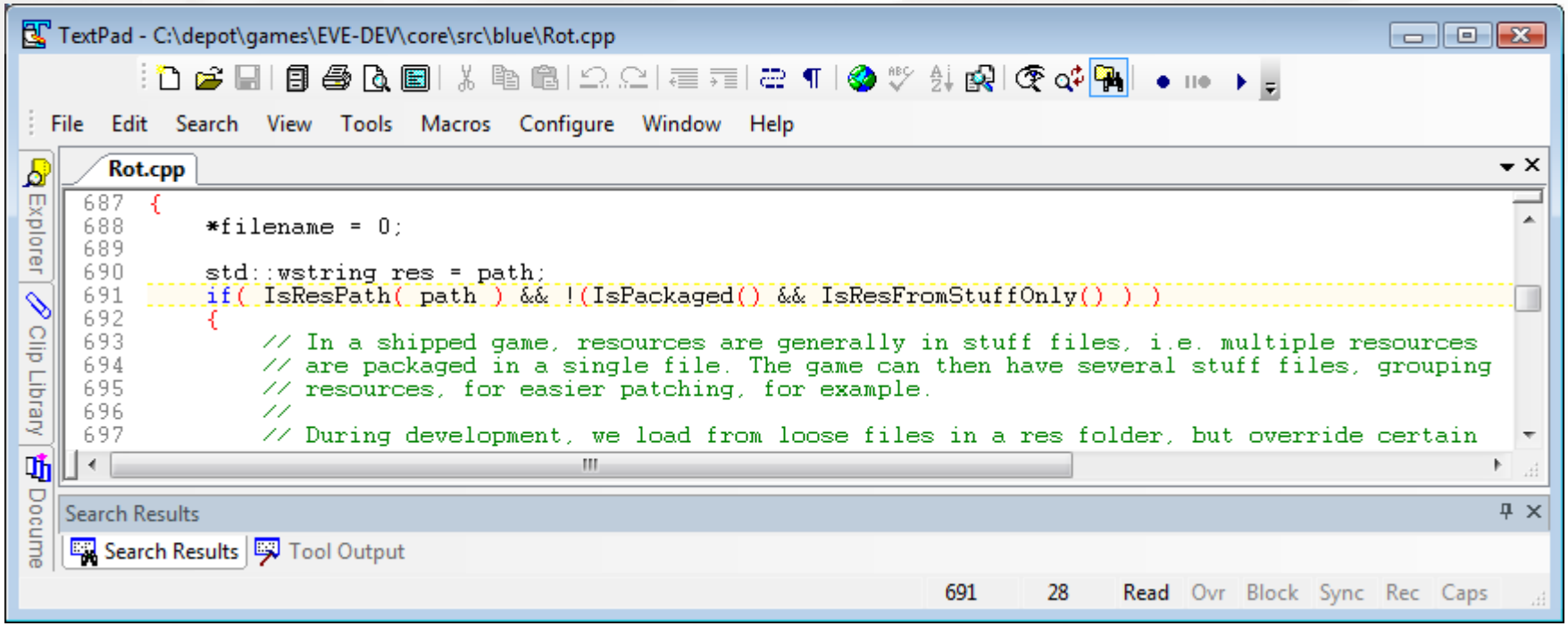
Release build == development build

- In Python:



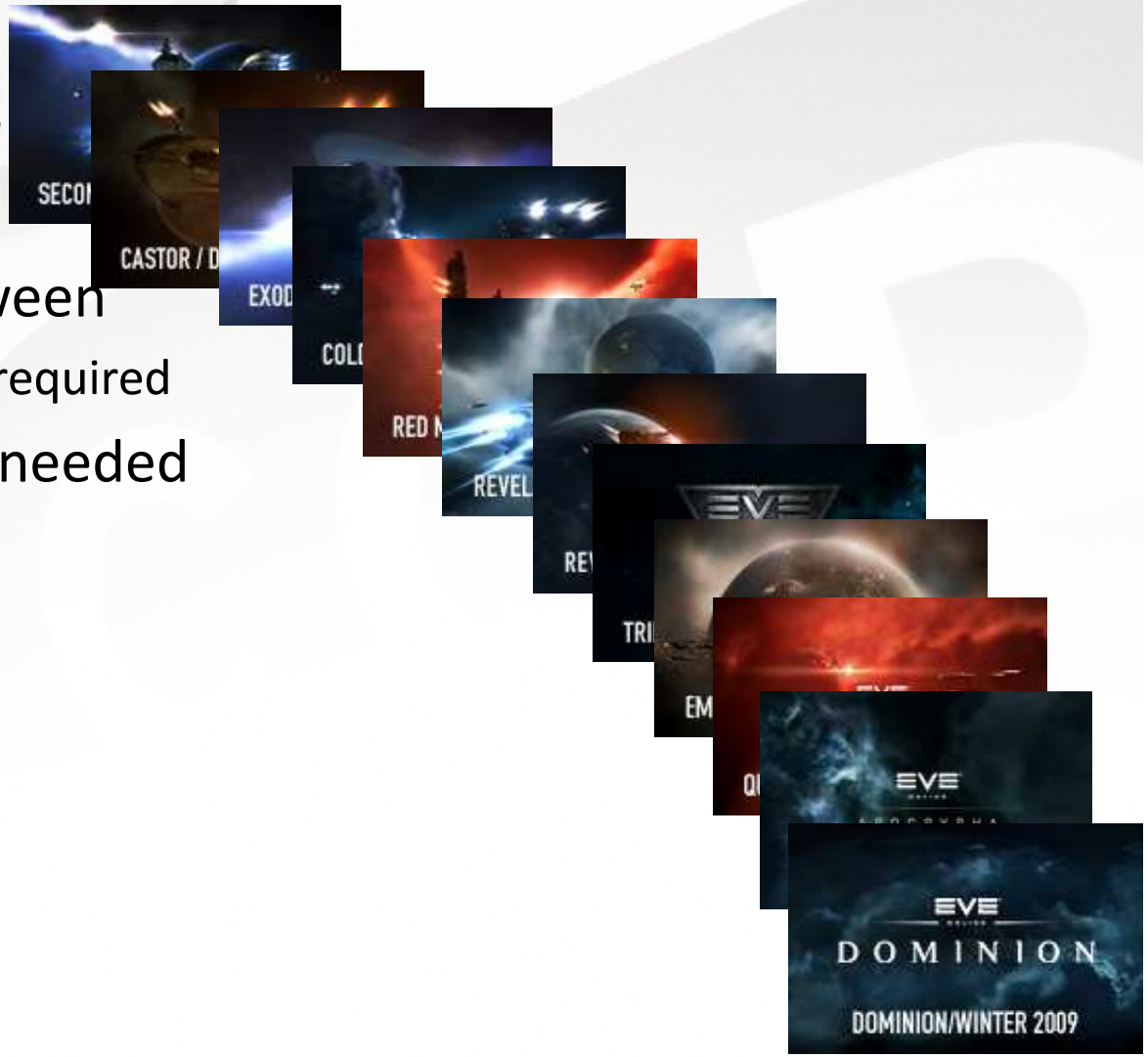
Release build == development build

- In C++:



Continuous releases

- 2 expansions per year
  - Working on the 13th
- Point releases in-between
  - Some optional, some required
- Server hotfixes when needed



Nurture the code

- Think ahead – simple, clear and clean
- Follow a coding standard
  - The question "why do it that way?", while valid to initiate a philosophical discussion, can easily be answered "because the guidelines say so".
- Think big – optimize
- Think carefully
- Balance features vs. technical debt

Nurture the developers

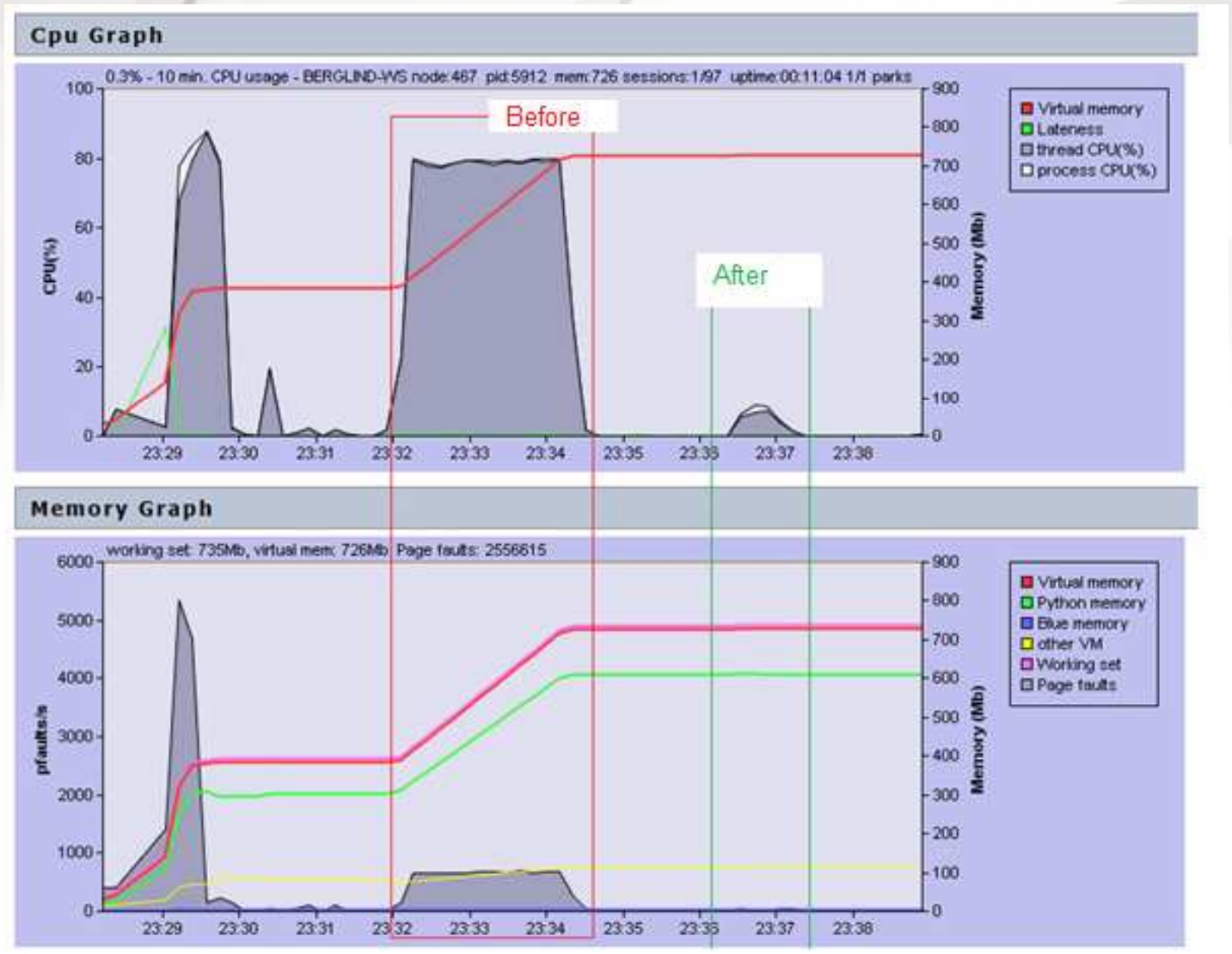
- Knowledge of the code base is very valuable
- Encourage knowledge transfer
- Manage requirements
- Sustainable pace

Success story - Mail

Mail

- Completely replaced existing system
  - And 3 others
- Migrated 150 million rows of data
- Strong emphasis on focus and scope
- Followup, improved design based on feedback
- Performance measurements

Mail


Cpu Graph
0.3% - 10 min. CPU usage - BERGLIND-WS node:467 pid:5912 mem:726 sessions:1/97 uptime:00:11:04 1/1 parks
Before
After
CPU(%)
Memory (Mb)
Virtual memory
Lateness
thread CPU(%)
process CPU(%)
23:29
23:30
23:31
23:32
23:33
23:34
23:35
23:36
23:37
23:38
Memory Graph
working set: 735Mb, virtual mem: 726Mb Page faults: 2556615
pfaults/s
Memory (Mb)
Virtual memory
Python memory
Blue memory
other VM
Working set
Page faults

Node-deaths after Trinity expansion

Node-deaths after Trinity expansion

- Random over a run, sometimes none, sometimes several
- Lots of log reading and code comparison
- Suspicious entries were spotted in logs
- Attached a debugger to a dying node on the live cluster
- Issue discovered to be a "wide" endless loop, i.e. in spawned tasklets
- Fixed logic in state machine:

  BAD:

  and not (self.enterDroneBay and self.activityState in [STATE_PURSUIT, STATE_DEPARTING])

  GOOD:

  and not self.enterDroneBay and not self.activityState in [STATE_PURSUIT, STATE_DEPARTING]

Performance issues after Apocrypha expansion

Performance issues after Apocrypha expansion

- Popular solar system could only handle a few hundred players
- Investigated from several angles at once
  - Start with an unaffected build and step through changes
  - Trace a particularly slow feature
  - Attach a debugger on a severely affected node on the live cluster
- The culprit turned out to be a sort in a critical location
- Removed one line:
  - dyingObjects.sort()

Questions

CCP Games
CP #2502

www.eveonline.com
berglind@ccpgames.com