# Technical Summary: Modernizing the 3DSteuerung System

1. Architecture Overview  
- Frontend: Qt 6 (PySide6) for GUI layout, docking panels, menu/toolbars  
- 3D Visualization: Panda3D embedded in Qt via shared window or shared memory  
- Backend: Python logic modules, modular backends for actuator communication  
- Data Flow: Joystick + user interaction -> Qt GUI -> motion compiler -> UDP or LinuxCNC -> actuator  
  
2. Component Breakdown  
  
a. Core Data Structures  
- Actuator: Holds ID, type (linear, rotary), limits, transform, backend binding  
- KinematicNode: Tree structure for defining forward/inverse kinematic chains  
- SceneObject: Panda3D node with transform and links to logical actuator model  
  
b. GUI System (Qt)  
- Dockable panels: left (actuator browser), right (property editor), bottom (timeline), center (Panda3D viewport)  
- Menu system: File, Tools, Diagnostics, Connect  
- Semi-transparent overlays: motion tracing, diagnostics, camera toolheads  
  
c. Visualization Layer (Panda3D)  
- Scenegraph representation of physical assembly  
- Support for real-time update of actuator state  
- Interactive editing: drag, rotate, attach, detach actuators  
  
3. Backend Communication  
  
a. UDP Backend  
- Sends setPos, setVel, setAcc etc. to actuators  
- Receives actPos, actVel, status, etc.  
- Supports watchdog, timestamps, feedback smoothing  
  
b. LinuxCNC Backend (Optional)  
- HAL component in user-space (Python or C)  
- Maps Qt signals to HAL pins and vice versa  
- Non-RT use recommended due to external RT loops in PLC/servo stack  
  
4. Joystick + Human Input  
- Python input polling for analog + digital inputs  
- Mapped via editable profiles to actuator degrees of freedom  
- Supports jog, setpoint streaming, velocity scaling  
  
5. Path Planning and Feasibility Analysis  
- G-code compatible parser with metadata extensions (speed zone, blending radius)  
- Motion segments compiled to spline-based motion plans  
- Physical feasibility checker:  
 - Velocity, acceleration, jerk limits per axis  
 - Blend/retime suggestions if constraints violated  
  
6. Simulation / Preview  
- Pre-execution visualization of planned motion  
- Time-synchronized playback with actual feedback overlay  
- Allows diagnostic comparison (planned vs actual path)  
  
7. Extensibility & Deployment  
- Plugin architecture for new actuator types, solvers, IO backends  
- Configurable project file format (e.g. JSON, XML)  
- Cross-platform: Windows + Linux  
  
8. Legacy Compatibility Goals  
- Full feature parity with wxPython legacy system  
- Improved maintainability, modularity, GUI responsiveness  
- Transition path for existing actuator configs and motion scripts  
  
Conclusion:  
This modular, Qt-based rearchitecture will form the foundation for future-proof high-performance kinematic systems with extensibility, visualization, and precision motion at its core.