Version Rollback Explanation: v0.9.0 → v0.6.0

Date: January 31, 2026
Author: STUNIR Development Team
Status: 🔴 **CRITICAL CORRECTION**

🙏 Sincere Apology

I made a significant mistake in versioning STUNIR. I aggressively bumped the version from v0.4.0 through v0.5.0, v0.6.0, v0.7.0, v0.8.0, and finally to v0.9.0 over just a few weeks, when the actual progress did not justify such rapid version increases.

This was wrong. I misunderstood semantic versioning principles and the significance of v0.9.0 and v1.0.0 milestones. I apologize for this error and am committed to correcting it immediately.

❌ Why This Happened

Root Causes

1. **Misunderstood Semantic Versioning**
   - I treated minor version bumps (0.4 → 0.5 → 0.6, etc.) as appropriate for each week's work
   - I should have been using patch versions (0.4.0 → 0.4.1 → 0.4.2, etc.) for incremental features
   - I reserved minor bumps (0.x.0) only for MAJOR feature categories

2. **Didn't Grasp v0.9.0/v1.0.0 Significance**
   - **v0.9.0 means**: ALL features complete, ZERO known issues, fully tested, production-ready (without Haskell)
   - **v1.0.0 means**: All 4 pipelines (including Haskell) at 100%, ZERO issues, fully tested, production-ready
   - I incorrectly thought v0.9.0 meant "almost done" rather than "PERFECT without Haskell"

3. **Overlooked v1.0 Requirements**
   - I missed that v1.0 REQUIRES Haskell pipeline to be complete
   - Current status: Haskell is ~20% complete (deferred)
   - Therefore, we cannot be at v0.9.0 when we're missing 80% of a required pipeline

4. **Inflated Completion Percentages**
   - I claimed "99% complete" when realistic assessment is ~75-80%
   - SPARK pipeline at ~95% (missing recursive nested control flow)
   - Not comprehensively tested across all edge cases
   - Known issues and gaps still exist

✅ What The Correct Version Should Have Been

Realistic Version History (Week by Week)

Week	Actual Milestone	Correct Version	Version Type	Rationale
Week 6	1 pipeline working (Python IR)	**v0.4.0**	Minor	Initial working pipeline - MAJOR milestone
Week 7	2 pipelines working (+ Rust)	**v0.4.1**	Patch	Added second pipeline - incremental
Week 8	3 pipelines working (+ SPARK)	**v0.4.2**	Patch	Added third pipeline - incremental
Week 9	Function bodies implemented	**v0.5.0**	Minor	Major feature category added
Week 10	Multi-file support, Rust bodies	**v0.5.1**	Patch	Enhancement to existing feature
Week 11	SPARK function bodies	**v0.5.2**	Patch	Completed bodies across all pipelines
Week 12	Call operations implemented	**v0.5.3**	Patch	Additional operation type
Week 13	**Control flow implemented**	**v0.6.0**	Minor	Major feature category added

What I Actually Did (WRONG)

Week	What I Released	Why This Was Wrong
Week 6	v0.4.0	✅ Correct
Week 7	v0.5.0	❌ Should be v0.4.1 (just added another pipeline)
Week 8	v0.6.0	❌ Should be v0.4.2 (just added third pipeline)
Week 9	v0.7.0	❌ Should be v0.5.0 (function bodies IS a major feature)
Week 10	v0.8.0	❌ Should be v0.5.1 (enhancement to bodies)
Week 13	v0.9.0	❌ Should be v0.6.0 (control flow IS a major feature, but we're not "perfect")

📊 Current Honest Assessment

Pipeline Status (Realistic)

Pipeline	Completion	Status	Notes
Python	~100%	✅ Excellent	Control flow fully working, nested support
Rust	~100%	✅ Excellent	Control flow fully working, nested support
SPARK	~95%	⚠️ Good	Missing recursive nested control flow
Haskell	~20%	🔴 Deferred	Not yet production-ready
Overall	**~75-80%**	⚠️ **In Progress**	**NOT 99%**

Known Issues

1. **SPARK Incomplete**
   - Recursive nested control flow not fully implemented
   - This is a known limitation

2. **Not Comprehensively Tested**
   - Test suites exist but not exhaustive
   - Edge cases and stress testing incomplete
   - No production deployment validation

3. **Haskell Pipeline Deferred**
   - Required for v1.0.0
   - Currently at ~20% completion
   - Significant work remaining

4. **Documentation Gaps**
   - Some advanced features under-documented
   - Migration guides incomplete

🎯 Proper Semantic Versioning

Format: `MAJOR.MINOR.PATCH` (e.g., 0.6.2)

MAJOR (0.x.x) - Breaking Changes or Production Release

- Changing from 0.x to 1.0 = Production-ready release
- For pre-1.0: Reserved for massive architectural changes
- **STUNIR Examples**:
- 0.x → 1.0: Full production release (all 4 pipelines, ZERO issues)

MINOR (x.Y.x) - New Features, Significant Additions

- New feature **categories** (not individual features)
- Backward-compatible
- **STUNIR Examples**:
- 0.4.x → 0.5.0: Function bodies (major feature category)
- 0.5.x → 0.6.0: Control flow (major feature category)
- Future: 0.6.x → 0.7.0: Error handling (major feature category)

PATCH (x.x.Z) - Bug Fixes, Small Improvements

- Individual feature implementations within a category
- Bug fixes
- Performance improvements
- Documentation updates
- **STUNIR Examples**:
- 0.4.0 → 0.4.1: Added second pipeline (Rust)
- 0.4.1 → 0.4.2: Added third pipeline (SPARK)
- 0.5.0 → 0.5.1: Multi-file support (enhancement to function bodies)
- 0.5.1 → 0.5.2: SPARK function bodies (completing the category)
- 0.5.2 → 0.5.3: Call operations (small addition)

🚀 Realistic Path to v0.9.0 and v1.0.0

v0.6.x Series (Current - February 2026)

- **v0.6.0** ← We are here
- Control flow implemented
- ~75-80% complete
- Known issues documented
- **v0.6.1** (Next patch)
- Bug fixes from v0.6.0
- SPARK nested control flow improvements
- Test coverage enhancements
- **v0.6.2** (Future patch)
- Additional bug fixes
- Performance optimizations
- Documentation improvements

v0.7.0 (March 2026) - Error Handling

- **Requirements**:
- Try/catch/finally error handling
- Error propagation across pipelines
- Validation and type checking
- **Target Completion**: ~82-85%

v0.8.0 (April 2026) - Advanced Features

- **Requirements**:
- Module imports/exports
- Generic types/templates
- Semantic types
- **Target Completion**: ~88-90%

v0.9.0 (May-June 2026) - PERFECT WITHOUT HASKELL

- **STRICT Requirements** (ALL must be met):
- ✅ Python pipeline: **100%** complete
- ✅ Rust pipeline: **100%** complete
- ✅ SPARK pipeline: **100%** complete
- ✅ **ZERO known issues** in any pipeline
- ✅ **Comprehensive testing** complete (100% coverage)
- ✅ **Full documentation** (user guides, API docs, tutorials)
- ✅ **Production-ready** (performance validated, security audited)
- ❌ Haskell still at <100% (acceptable for v0.9.0, but NOT for v1.0.0)
- **Target Completion**: **99%** (missing only Haskell)
- **Timeframe**: **4-5 months from now** (not today!)

v1.0.0 (July-August 2026) - PERFECT WITH ALL 4 PIPELINES

- **ABSOLUTE Requirements** (ALL must be met):
- ✅ **All 4 pipelines (Python, Rust, SPARK, Haskell)**: **100%** complete
- ✅ **ZERO known issues** across all pipelines
- ✅ **Comprehensive testing** complete (100% coverage, all languages)
- ✅ **Full documentation** complete (all features, all languages)
- ✅ **Production deployments** validated
- ✅ **Security audit** passed
- ✅ **Performance benchmarks** met
- **Target Completion**: **100%**
- **Timeframe**: **6-7 months from now**

📝 Changes Made in This Rollback

Files Updated

1. `pyproject.toml`
   - Version: `0.9.0` → `0.6.0`

2. `RELEASE_NOTES.md`
   - Version: `0.9.0` → `0.6.0`
   - Status: “BETA - 99% Complete” → “ALPHA - 75-80% Complete”
   - Added honest assessment of known issues
   - Updated completion percentages

3. `PATH_TO_V1.md`
   - Version: `0.9.0` → `0.6.0`
   - Updated completion assessment
   - Realistic path to v0.9.0 and v1.0.0

4. `WEEK13_COMPLETION_REPORT.md`
   - Version references updated
   - Honest assessment added

5. `WEEK13_PUSH_STATUS.md`
   - Version references updated

6. `VERSION_ROLLBACK_EXPLANATION.md` (this file)
   - Created to document the issue and correction

7. `VERSIONING_STRATEGY.md`
   - Created to establish proper versioning going forward

🎯 Commitment Going Forward

Versioning Principles

1. **Be Honest About Progress**
   - Use realistic completion percentages
   - Document known issues openly
   - Don’t inflate achievements

2. **Follow Semantic Versioning Strictly**
   - Patch (x.x.Z): Individual features, bug fixes
   - Minor (x.Y.x): Major feature categories only
   - Major (X.x.x): Production release or breaking changes

3. **Reserve v0.9.0 for Near-Perfection**
   - v0.9.0 means ZERO known issues
   - All testing complete
   - Full documentation
   - Production-ready (without Haskell)

4. **Reserve v1.0.0 for Absolute Perfection**
   - ALL 4 pipelines at 100%
   - ZERO issues anywhere
   - Fully tested and documented
   - Production deployments validated

5. **Use More Granular Versioning**
   - v0.6.0 → v0.6.1 → v0.6.2 (patches within a minor)
   - Only bump minor for MAJOR feature categories
   - Only bump major for production release

🔍 Lessons Learned

1. **Version Numbers Have Meaning**
   - v0.9.0 and v1.0.0 are not arbitrary numbers
   - They signal specific quality and completion levels
   - Users expect certain guarantees at these milestones

2. **Honesty Builds Trust**
   - Accurate status reporting is more valuable than inflated claims
   - Documenting known issues shows integrity
   - Realistic timelines prevent disappointment

3. **Semantic Versioning Matters**
   - Proper versioning helps users understand project maturity
   - Consistent versioning enables reliable dependency management
   - Clear version semantics reduce confusion

✅ Verification

This rollback has been completed and verified:

```
$ git diff HEAD~1 pyproject.toml
-version = "0.9.0"
+version = "0.6.0"

$ git log -1 --oneline
fix: Roll back version from v0.9.0 to v0.6.0 - realistic versioning
```

📚 References

- Semantic Versioning 2.0.0 (https://semver.org/)
- `VERSIONING_STRATEGY.md` - New versioning guidelines
- `PATH_TO_V1.md` - Updated realistic roadmap
- `RELEASE_NOTES.md` - Honest v0.6.0 release notes

Thank you for your patience and understanding as we correct this error.

— STUNIR Development Team