# STUNIR Versioning Strategy

**Status**: ✅ **ACTIVE POLICY**
**Effective Date**: January 31, 2026
**Last Updated**: January 31, 2026
**Owner**: STUNIR Development Team

---

## Overview

This document defines the official versioning strategy for STUNIR, following Semantic Versioning 2.0.0 (https://semver.org/) principles adapted for pre-1.0 development.

**Current Version**: `0.6.0`
**Target v1.0.0 Date**: July-August 2026 (6-7 months)

---

## Semantic Versioning Format

```
MAJOR.MINOR.PATCH
  │     │     │
  │     │     └──── Patch: Bug fixes, small features, documentation
  │     └────────── Minor: New feature categories, significant additions
  └──────────────── Major: Production release (1.0) or breaking changes
```

### Version Number Significance

- **0.x.x** = Pre-production development
- **1.0.0** = First production-ready release
- **1.x.x** = Production with backward-compatible additions
- **2.0.0+** = Breaking changes in production

---

## STUNIR Version Semantics

### MAJOR Version (X.y.z)

**When to increment:**
- First production release: `0.x.x → 1.0.0`
- Breaking changes in production: `1.x.x → 2.0.0`
- Major architectural overhauls (post-1.0 only)

**Requirements for MAJOR version 1.0.0:**
- ✅ All 4 pipelines (Python, Rust, SPARK, **Haskell**) at **100%**
- ✅ **ZERO known critical or high-severity issues**
- ✅ **Comprehensive testing** (unit, integration, end-to-end)
- ✅ **Full documentation** (user guides, API docs, tutorials, migration guides)

- ✅ **Production deployments** validated
- ✅ **Security audit** completed
- ✅ **Performance benchmarks** met

**STUNIR 1.0.0 is NOT NEGOTIABLE** - all criteria must be met.

---

## MINOR Version (x.Y.z)

**When to increment:**
- Adding a **major feature category** (not individual features)
- Significant capability expansion
- Backward-compatible

**What constitutes a “major feature category”:**
- ✅ Control flow (if/while/for) - This is ONE category
- ✅ Error handling (try/catch/finally) - This is ONE category
- ✅ Module system (imports/exports) - This is ONE category
- ✅ Generic types/templates - This is ONE category
- ✅ Async/await support - This is ONE category

**What does NOT constitute a major feature category:**
- ❌ Adding one more pipeline (use patch)
- ❌ Adding one operation type (use patch)
- ❌ Bug fixes (use patch)
- ❌ Performance improvements (use patch)
- ❌ Documentation updates (use patch)

**STUNIR Examples:**
- `0.4.x → 0.5.0`: Function bodies (major category)
- `0.5.x → 0.6.0`: Control flow (major category)
- `0.6.x → 0.7.0`: Error handling (major category)
- `0.7.x → 0.8.0`: Module system (major category)

---

## PATCH Version (x.y.Z)

**When to increment:**
- Individual feature implementations within a category
- Bug fixes (any severity)
- Performance improvements
- Documentation updates
- Test coverage improvements
- Refactoring without feature changes

**STUNIR Examples:**
- `0.4.0 → 0.4.1`: Added Rust pipeline (incremental)
- `0.4.1 → 0.4.2`: Added SPARK pipeline (incremental)
- `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 (individual operation)

- `0.6.0 → 0.6.1`: Bug fixes from v0.6.0
- `0.6.1 → 0.6.2`: SPARK nested control flow improvements

---

## Version Milestone Definitions

### v0.9.0 - "Near-Perfect Without Haskell"

**Status**: Future milestone (target: May-June 2026)

**STRICT Requirements** (ALL must be met):
1. ✅ Python pipeline: **100%** complete
2. ✅ Rust pipeline: **100%** complete
3. ✅ SPARK pipeline: **100%** complete
4. ✅ **ZERO known critical or high-severity issues**
5. ✅ **Comprehensive testing** (95%+ coverage)
6. ✅ **Full documentation** (all features documented)
7. ✅ **Production-ready** (performance, security validated)
8. ⚠️ Haskell pipeline may be <100% (acceptable for v0.9.0)

**Quality Bar:**
- No known crashes
- No data corruption bugs
- No security vulnerabilities
- All edge cases tested
- Performance meets benchmarks
- Documentation complete

**Timeframe**: 4-5 months from v0.6.0

---

### v1.0.0 - "Production Release"

**Status**: Future milestone (target: July-August 2026)

**ABSOLUTE Requirements** (ALL must be met):
1. ✅ **All 4 pipelines** (Python, Rust, SPARK, **Haskell**): **100%**
2. ✅ **ZERO known issues** (any severity)
3. ✅ **Comprehensive testing** (100% coverage, all languages)
4. ✅ **Full documentation** (complete, all languages)
5. ✅ **Production deployments** validated in real-world scenarios
6. ✅ **Security audit** passed (independent review)
7. ✅ **Performance benchmarks** met (all targets achieved)
8. ✅ **Migration guides** complete (from all previous versions)
9. ✅ **Support infrastructure** ready (issue tracking, forums, etc.)
10. ✅ **Release artifacts** signed and published

**Quality Bar:**
- Absolutely NO known bugs
- Zero security vulnerabilities
- 100% feature parity across all pipelines
- Deterministic builds verified

- Cross-platform validated
- Long-term support commitment

**Timeframe**: 6-7 months from v0.6.0

---

## Version Bump Decision Tree

```
START: What changed?
│
├─ Is this the v1.0.0 production release?
│  └─ YES → Bump MAJOR (0.x.x → 1.0.0)
│
├─ Did we add a MAJOR FEATURE CATEGORY?
│  ├─ YES → Bump MINOR (0.6.x → 0.7.0)
│  └─ NO → Continue
│
├─ Did we add an individual feature/fix/improvement?
│  └─ YES → Bump PATCH (0.6.0 → 0.6.1)
│
└─ Just documentation/comments?
   └─ Bump PATCH (0.6.0 → 0.6.1)
```

### Examples

| Change | Old Version | New Version | Rationale |
|---|---|---|---|
| Added control flow | 0.5.3 | **0.6.0** | Major feature category |
| Fixed SPARK bug | 0.6.0 | **0.6.1** | Bug fix (patch) |
| Added fourth pipeline | 0.6.1 | **0.6.2** | Individual feature (patch) |
| Improved docs | 0.6.2 | **0.6.3** | Documentation (patch) |
| Added error handling | 0.6.3 | **0.7.0** | Major feature category |
| Fixed memory leak | 0.7.0 | **0.7.1** | Bug fix (patch) |
| Performance optimization | 0.7.1 | **0.7.2** | Improvement (patch) |
| Added module system | 0.7.2 | **0.8.0** | Major feature category |
| All requirements met | 0.9.x | **1.0.0** | Production release |

---

## Release Process

### 1. Pre-Release Checklist

Before bumping version:
- [ ] All tests passing
- [ ] No known critical bugs
- [ ] Documentation updated
- [ ] CHANGELOG.md updated
- [ ] RELEASE_NOTES.md updated

### 2. Version Bump

```
# Update version in pyproject.toml
version = "0.6.1"

# Update RELEASE_NOTES.md
## Version 0.6.1 - February X, 2026

# Update PATH_TO_V1.md
**Current Version**: 0.6.1
```

### 3. Git Tagging

```
# Commit version bump
git add pyproject.toml RELEASE_NOTES.md PATH_TO_V1.md
git commit -m "chore: Bump version to v0.6.1"

# Create annotated tag
git tag -a v0.6.1 -m "Release v0.6.1: Bug fixes and improvements"

# Push with tags
git push origin devsite --tags
```

### 4. Release Notes

- Summarize changes clearly
- List known issues
- Provide migration guidance if needed
- Update completion percentage honestly

---

## Versioning Anti-Patterns (What NOT to Do)

### ❌ DON’T: Bump minor version for every week’s work

```
Week 1: v0.4.0
Week 2: v0.5.0  ← WRONG
Week 3: v0.6.0  ← WRONG
```

### ✅ DO: Use patch versions for incremental work

```
Week 1: v0.4.0
Week 2: v0.4.1  ← Correct
Week 3: v0.4.2  ← Correct
Week 4: v0.5.0  ← Correct (if major feature category added)
```

### ❌ DON’T: Jump to v0.9.0 prematurely

- v0.9.0 means “near-perfect”
- Known issues = NOT v0.9.0
- Missing features = NOT v0.9.0
- Incomplete testing = NOT v0.9.0

### ✅ DO: Be honest about completion

- Document known issues openly
- Use realistic completion percentages
- Reserve v0.9.0 for true near-perfection

### ❌ DON’T: Inflate completion percentages

```
"99% complete" when actually 75% ← WRONG
```

### ✅ DO: Provide honest assessments

```
"~75-80% complete - significant work remains" ← Correct
```

---

## Completion Percentage Guidelines

### How to Calculate Realistic Completion

1. **Pipeline Status** (40% weight)
   - Python: x%
   - Rust: x%
   - SPARK: x%
   - Haskell: x%
   - Average: (sum / 4)

2. **Feature Completeness** (30% weight)
   - Core IR generation: x%
   - Code emission: x%
   - Control flow: x%
   - Error handling: x%
   - Module system: x%
   - Average: (sum / feature_count)

3. **Quality Metrics** (30% weight)
   - Test coverage: x%
   - Known issues: (none=100%, critical=-20%, high=-10%, medium=-5%)

   - Documentation: x%
   - Average: (sum / 3)

4. **Overall** = (Pipeline * 0.4) + (Features * 0.3) + (Quality * 0.3)

## Current v0.6.0 Assessment

```
Pipeline Status:
- Python: ~100% → 25/25 points
- Rust: ~100% → 25/25 points
- SPARK: ~95% → 24/25 points
- Haskell: ~20% → 5/25 points
Subtotal: 79/100 → 79% * 0.4 = 31.6%

Feature Completeness:
- IR generation: 100%
- Code emission: 100%
- Function bodies: 100%
- Control flow: 95% (SPARK partial)
- Error handling: 0%
- Module system: 0%
Subtotal: 395/600 → 65.8% * 0.3 = 19.7%

Quality Metrics:
- Test coverage: ~70%
- Known issues: Medium severity → -5%
- Documentation: ~80%
Subtotal: (70 + 95 + 80) / 3 = 81.7% * 0.3 = 24.5%

Overall: 31.6% + 19.7% + 24.5% = 75.8%
```

**Result**: ~75-80% complete (realistic assessment)

---

## Version Planning

### Roadmap to v1.0.0

| Version | Target Date | Focus | Completion |
|---|---|---|---|
| **v0.6.0** | Jan 31, 2026 | ✅ Control flow | ~75-80% |
| **v0.6.1** | Feb 7, 2026 | Bug fixes, SPARK improvements | ~77% |
| **v0.6.2** | Feb 14, 2026 | Additional bug fixes | ~78% |
| **v0.7.0** | Mar 7, 2026 | Error handling (try/catch) | ~82% |
| **v0.7.1** | Mar 14, 2026 | Error handling refinements | ~83% |
| **v0.8.0** | Apr 4, 2026 | Module system (imports/exports) | ~88% |
| **v0.8.1** | Apr 11, 2026 | Module refinements | ~89% |
| **v0.8.2** | Apr 18, 2026 | Generic types/templates | ~90% |
| **v0.9.0** | May 23, 2026 | **All 3 pipelines perfect** | **~99%** |
| **v0.9.1** | Jun 6, 2026 | Haskell progress | ~99.5% |
| **v1.0.0** | Jul 18, 2026 | 🎉 **PRODUCTION RELEASE** | **100%** |

---

## Quality Gates for Key Milestones

### v0.9.0 Quality Gate

Must pass ALL checks:
- [ ] Python pipeline: 100% feature complete
- [ ] Rust pipeline: 100% feature complete
- [ ] SPARK pipeline: 100% feature complete
- [ ] Zero critical/high severity bugs
- [ ] Test coverage >95%
- [ ] All documentation complete
- [ ] Performance benchmarks met
- [ ] Security scan passed

**If ANY check fails → NOT v0.9.0**

### v1.0.0 Quality Gate

Must pass ALL checks:
- [ ] All v0.9.0 checks passed
- [ ] Haskell pipeline: 100% feature complete
- [ ] Zero bugs (any severity)
- [ ] Test coverage 100%
- [ ] Production deployments validated
- [ ] Independent security audit passed
- [ ] Long-term support plan finalized
- [ ] Release artifacts signed

**If ANY check fails → NOT v1.0.0**

---

## Version Update Checklist

When bumping version, update these files:

### Required Updates

- [ ] `pyproject.toml` - version field
- [ ] `RELEASE_NOTES.md` - new section at top
- [ ] `PATH_TO_V1.md` - current version field
- [ ] `CHANGELOG.md` - update links and version references

### Optional Updates (if referenced)

- [ ] `README.md` - if version mentioned
- [ ] `docs/WEEK*_COMPLETION_REPORT.md` - completion reports
- [ ] `*_PUSH_STATUS.md` - push status documents

### Git Operations

- [ ] Commit with message: `chore: Bump version to vX.Y.Z`
- [ ] Tag with annotated tag: `git tag -a vX.Y.Z -m "Release vX.Y.Z"`
- [ ] Push with tags: `git push origin devsite --tags`

---

## References

- Semantic Versioning 2.0.0 (https://semver.org/)
- `VERSION_ROLLBACK_EXPLANATION.md` - Why we rolled back from v0.9.0
- `PATH_TO_V1.md` - Detailed roadmap to v1.0.0
- `RELEASE_NOTES.md` - Historical release notes

---

## Document History

| Version | Date | Changes |
|---|---|---|
| 1.0 | Jan 31, 2026 | Initial version after v0.9.0 roll-back |

---

**Questions?** Contact the STUNIR development team or refer to `VERSION_ROLLBACK_EXPLANATION.md`.

**Remember**: Version numbers are commitments to quality. Use them wisely.