Based on my analysis of the code changes, here are the key modifications and their potential impacts:

1. Asynchronous Strategy Creation:
2. Changed `Strategy::SendCreateRequest` to `Strategy::SendAsyncCreateRequest` for hedge order strategies
3. Added handling of asynchronous strategy creation replies in new `HandleStrategyRequestReply` method
4. Potential impact: Improved performance by reducing latency in hedge order creation
5. Refactoring of Hedge Request Handling:
6. Introduced new `m_hedgeRequests` map to track pending asynchronous hedge requests
7. Moved logic for subscribing to strategies and starting hedge timers to `HandleStrategyRequestReply`
8. Potential impact: Better organization of hedge request lifecycle, supports async creation
9. Terminology Changes:
10. Renamed "Immediate" to "Initial" in several method names and comments
11. Example: `DeletedImmediateHedgeOrder -> DeletedInitialHedgeOrder`
12. Potential impact: Improved clarity in code, no functional change
13. Minor Logic Adjustments:
14. Changed condition `m_skipInitialIOC` to `m_skipInitialHedge`
15. Updated `HasHedges()` method to use new `HasHedgeOrders()` helper
16. Potential impact: Slight changes in hedge order creation logic
17. New Helper Methods:
18. Added `HasHedgeOrders()` method to check for active hedge orders or requests
19. Potential impact: Encapsulates hedge order status checking, improves code readability

Critical Findings:

1. Asynchronous Strategy Creation:
2. This is the most significant change and could impact the timing and order of hedge creation

3. Ensure proper error handling and timeout mechanisms are in place for async requests
4. Verify that the system can handle scenarios where async replies come in unexpected order
5. State Management:
6. With the introduction of `m_hedgeRequests`, ensure proper cleanup of this map to avoid memory leaks
7. Verify that all edge cases (e.g., cancellations, failures) are handled correctly in the async flow
8. Regression Risks:
9. The changes to hedge order creation logic (e.g., `m_skipInitialHedge`) may affect existing behavior
10. Thoroughly test different hedge scenarios to ensure no unintended side effects
11. Performance Impact:
12. While async creation should improve performance, monitor system behavior under high load
13. Verify that the async approach doesn't lead to resource contention or oversubscription

Recommendations:

1. Implement comprehensive unit and integration tests for the new async hedge order creation flow
2. Add logging or tracing to track the lifecycle of async hedge requests for debugging
3. Review and update any documentation or comments related to the hedge order creation process
4. Consider adding metrics to monitor the performance impact of these changes in production
5. Ensure that error handling and recovery mechanisms are robust for the new async flow

Overall, these changes appear to be aimed at improving performance and code organization for hedge order creation in the spread quoter system. The shift to asynchronous strategy creation is the most significant change and warrants careful testing and monitoring.