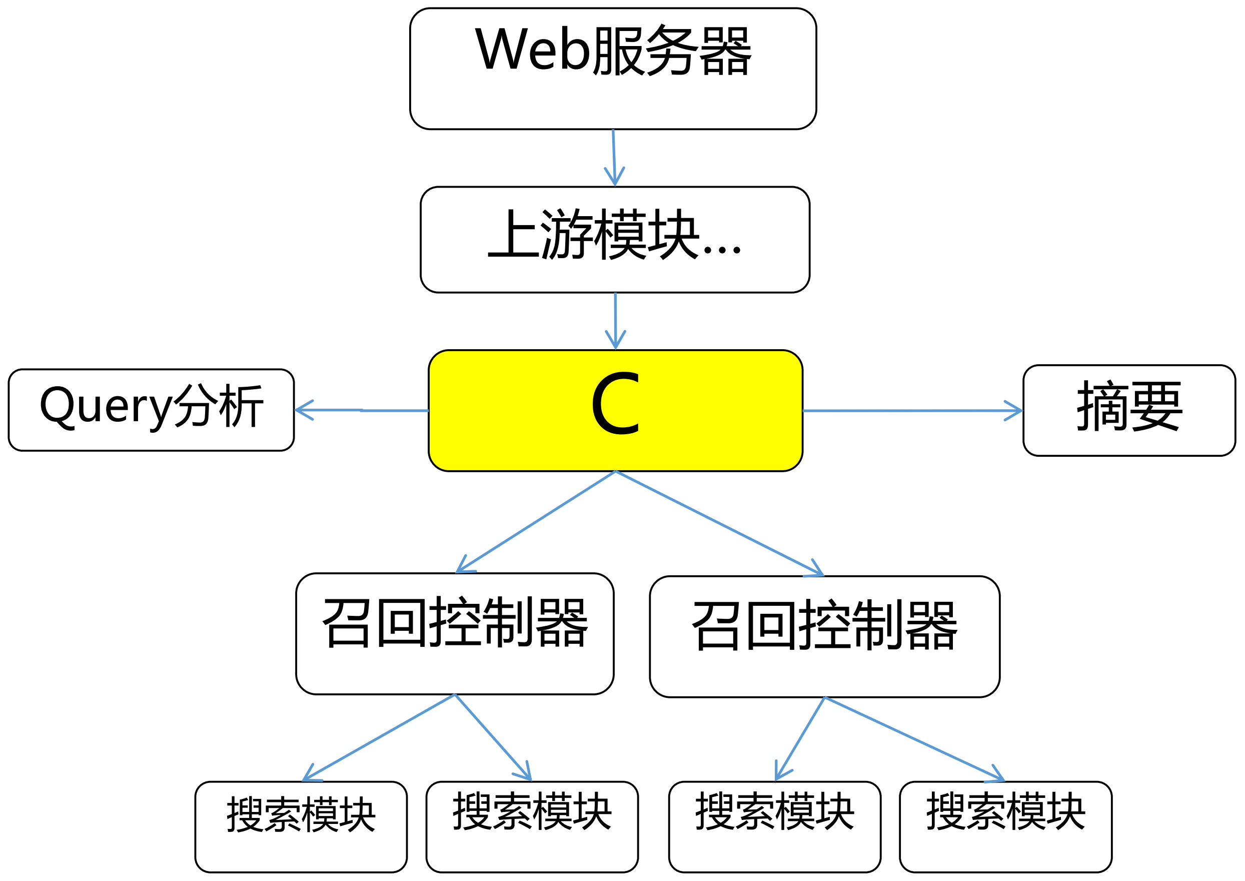
网页搜索核心模块重构 2014.6~至今

网页搜索部 & 工程效率部

冯上 马波

C模块 - 位置



模块C

- 多产品支持
  - 网页搜索
  - 移动搜索
  - 知道/贴吧/文库
- 开发密集
  - 每周上线10-20个功能
  - 最近1.5年有206人贡献过代码

内容简介

项目背景

阶段性成果

我们如何做 设计层面、编码层面、工具层面

项目背景

模块接连出现上线回滚 一个月6次

代码复杂难以维护 复杂度高

系统性错误难以排查 内存、指针、core

项目阶段性成果

Query分析交互部分全部OO化

面向对象的设计，接口清晰，职责明确

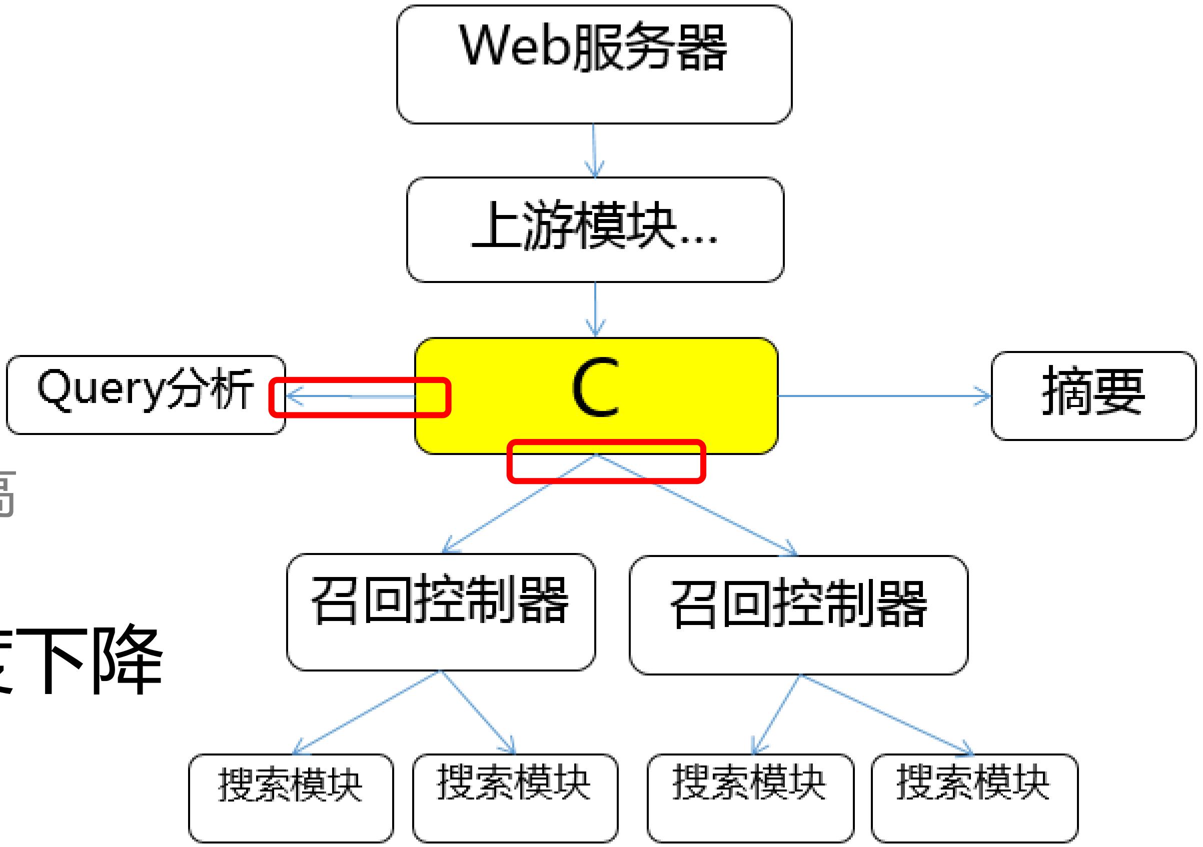
搜索控制器交互状态机重新设计

逻辑清晰化，复杂度降低，内存下降，性能提高

单元测试覆盖率提升、代码复杂度下降

类平均大小降低25%，平均圈复杂度下降10%

清除无用代码 2.6W+ lines



我们如何做

指导思想 小步快跑、演进式设计、SOLID原则

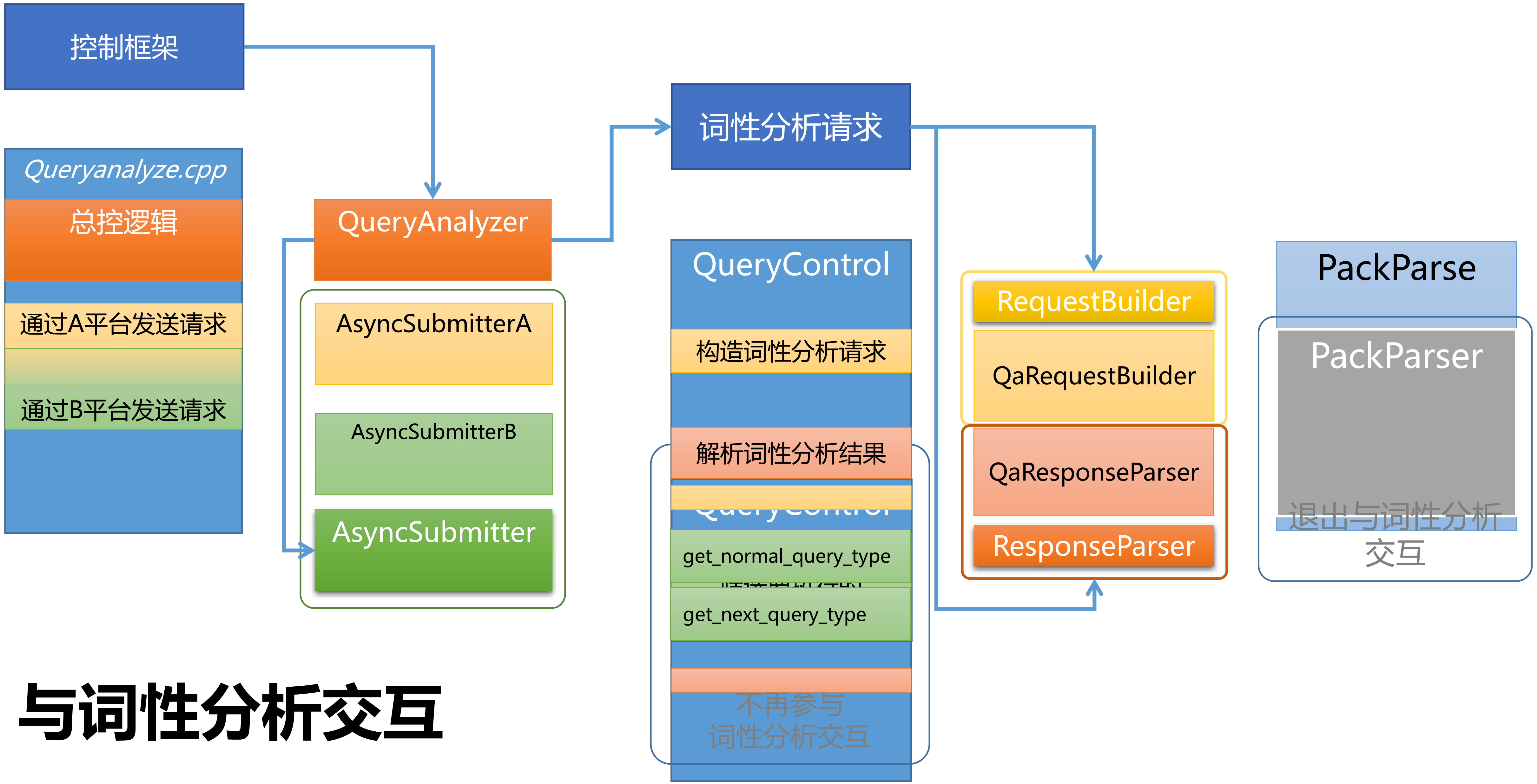
实现细节 设计层面、编码层面、工具层面

设计层面

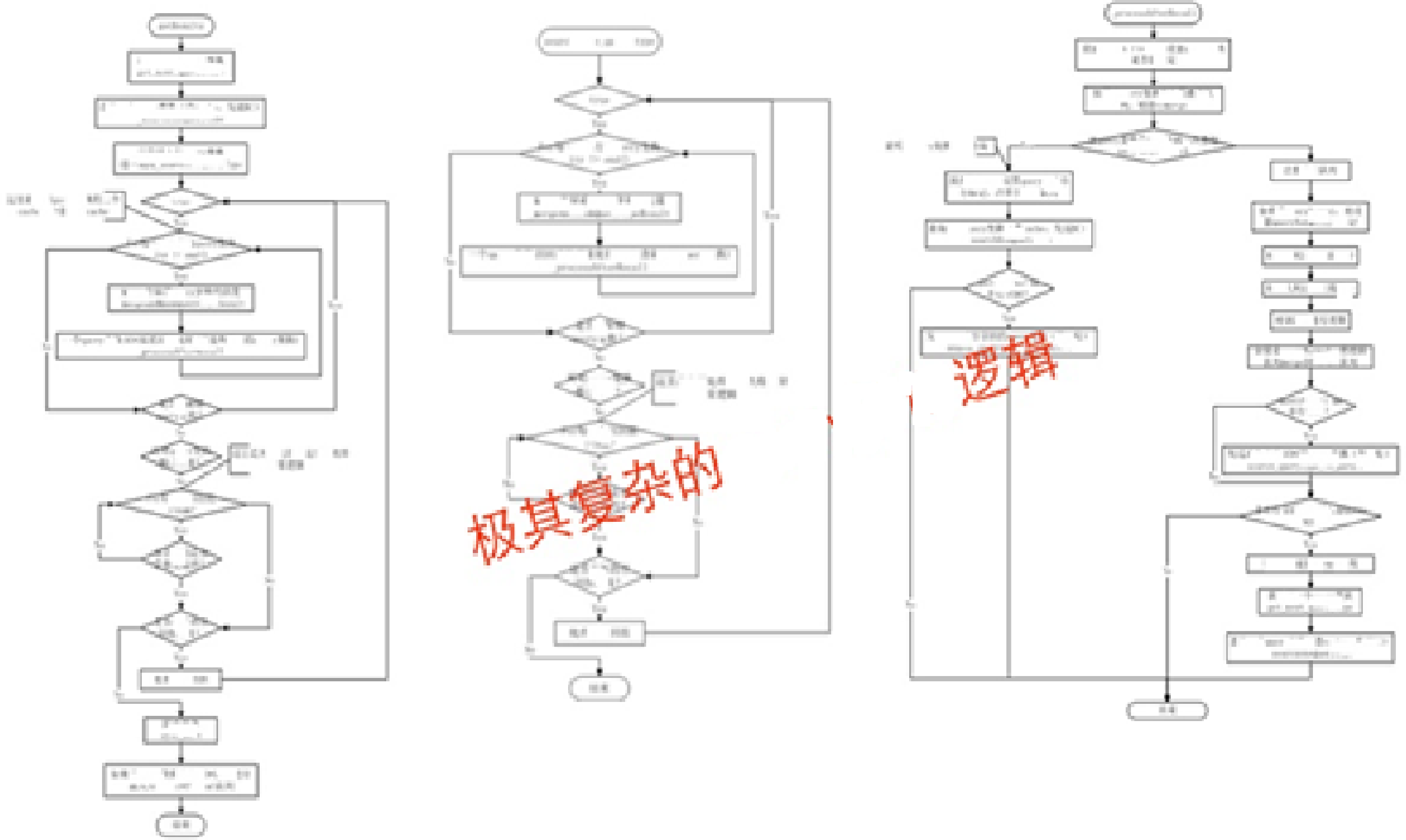
架构：混乱/messy → 清晰/clear and focused

初级OO/basic OO → 设计模式/design pattern

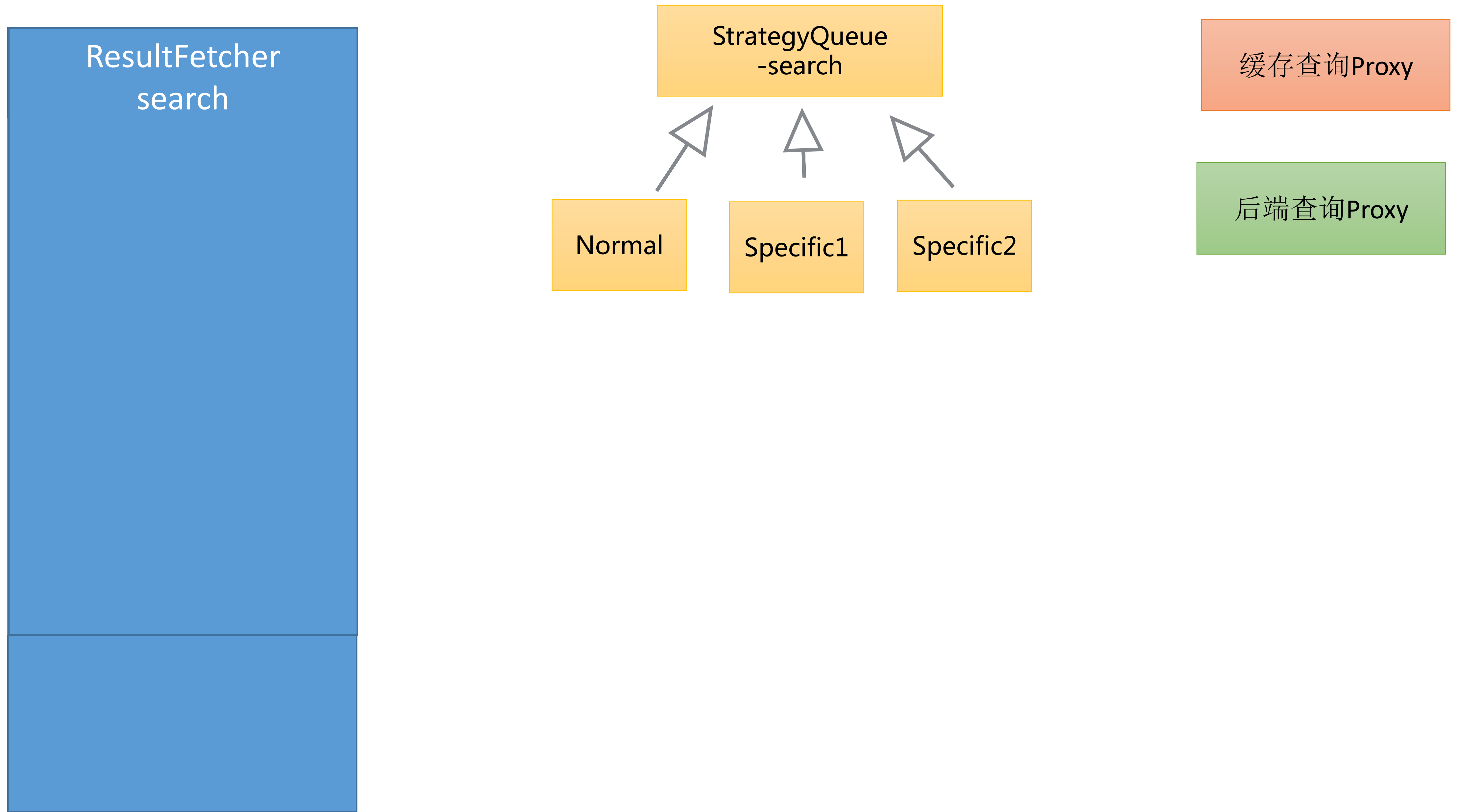
架构：混乱/messy → 清晰/clear and focused

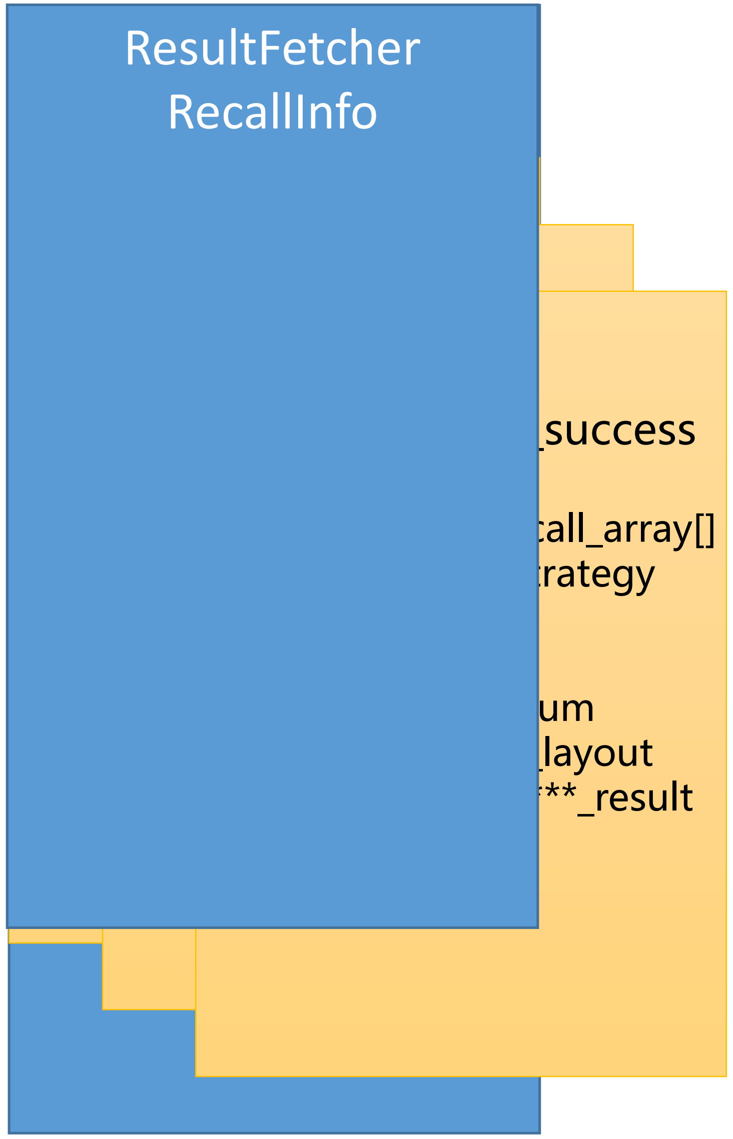

控制框架
Queryanalyze.cpp
总控逻辑
通过A平台发送请求
通过B平台发送请求
QueryAnalyzer
AsyncSubmitterA
AsyncSubmitterB
AsyncSubmitter
词性分析请求
QueryControl
构造词性分析请求
解析词性分析结果
get_normal_query_type
get_next_query_type
不再参与
词性分析交互
RequestBuilder
QaRequestBuilder
QaResponseParser
ResponseParser
PackParse
PackParser
退出与词性分析
交互

与词性分析交互

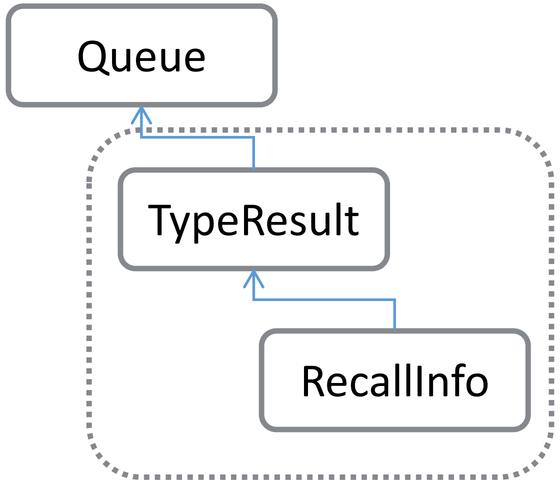


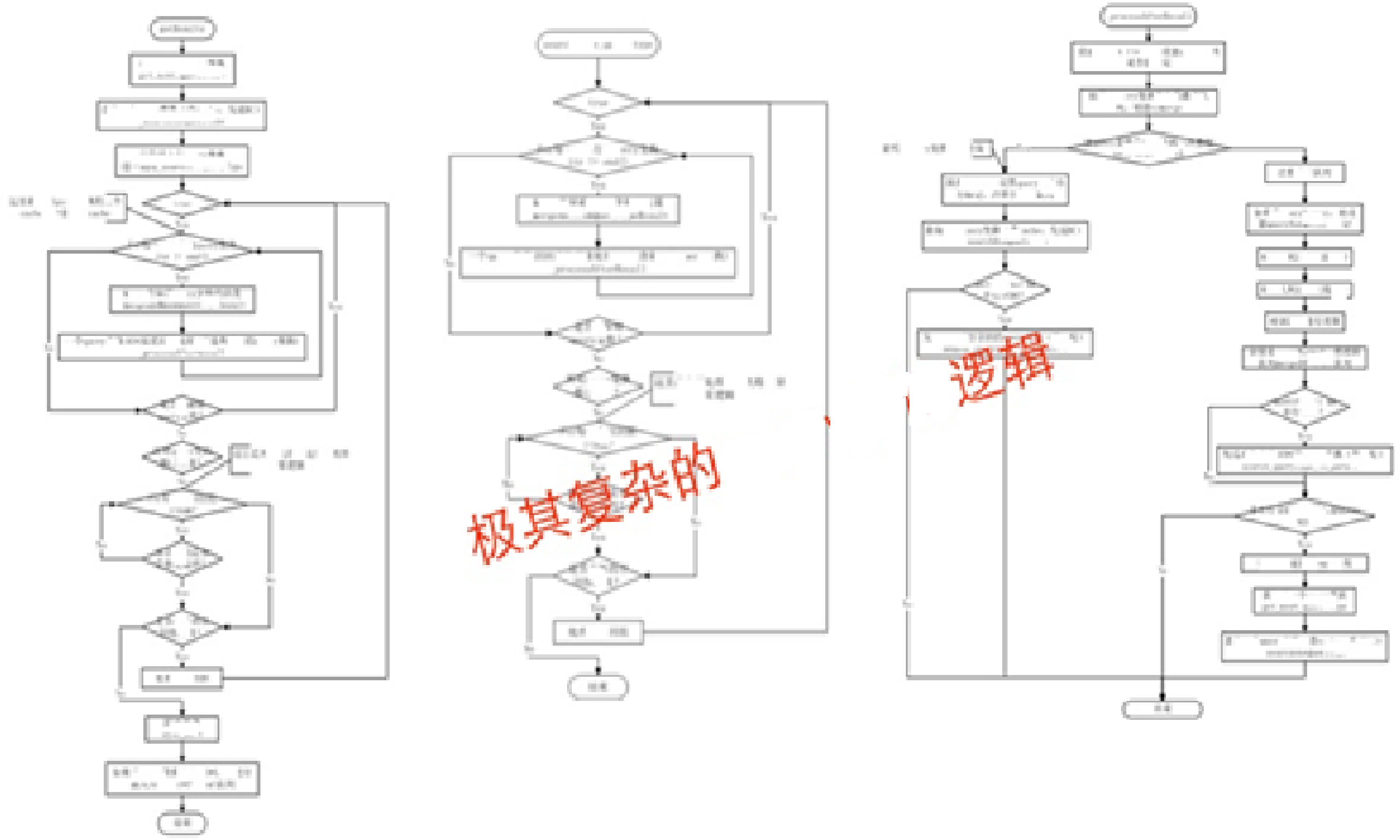
极其复杂的 逻辑

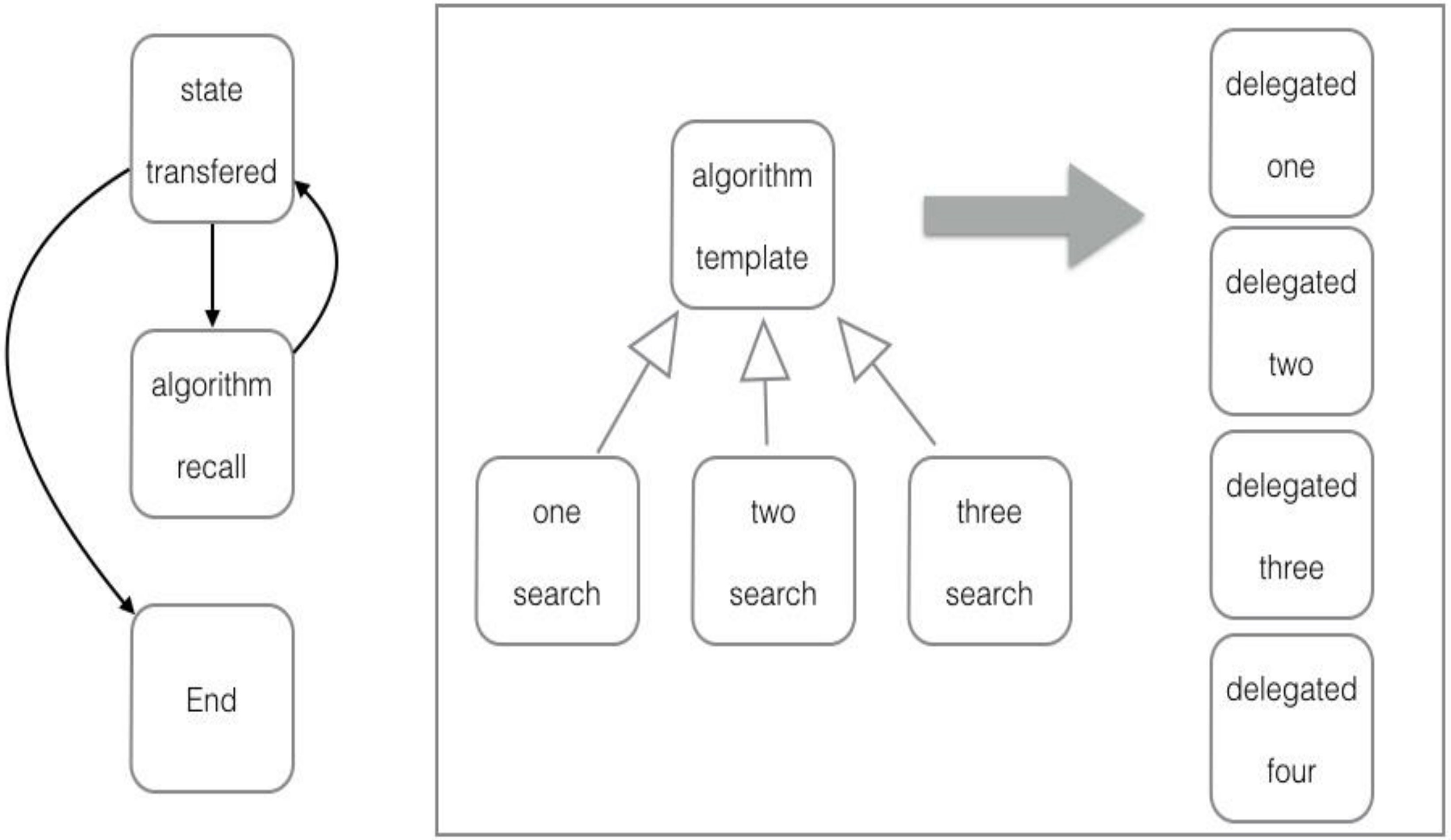

ResultFetcher
search
StrategyQueue
-search
Normal
Specific1
Specific2
缓存查询Proxy
后端查询Proxy



RecallInfo
-finish

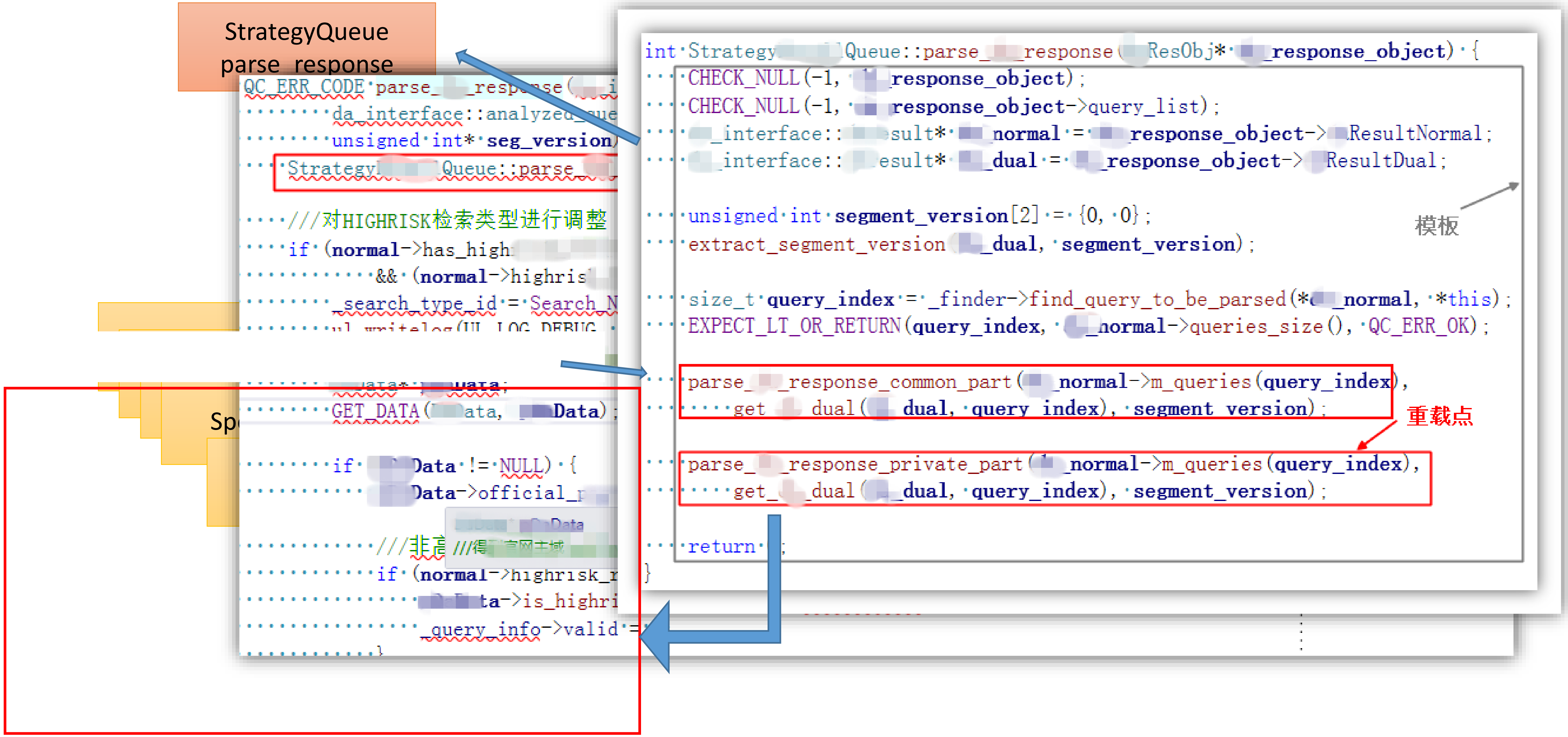



极其复杂的
逻辑


state transfered
algorithm recall
End
algorithm template
one search
two search
three search
delegated one
delegated two
delegated three
delegated four

初级OO/basic OO→ **设计模式**/design pattern

Template Method

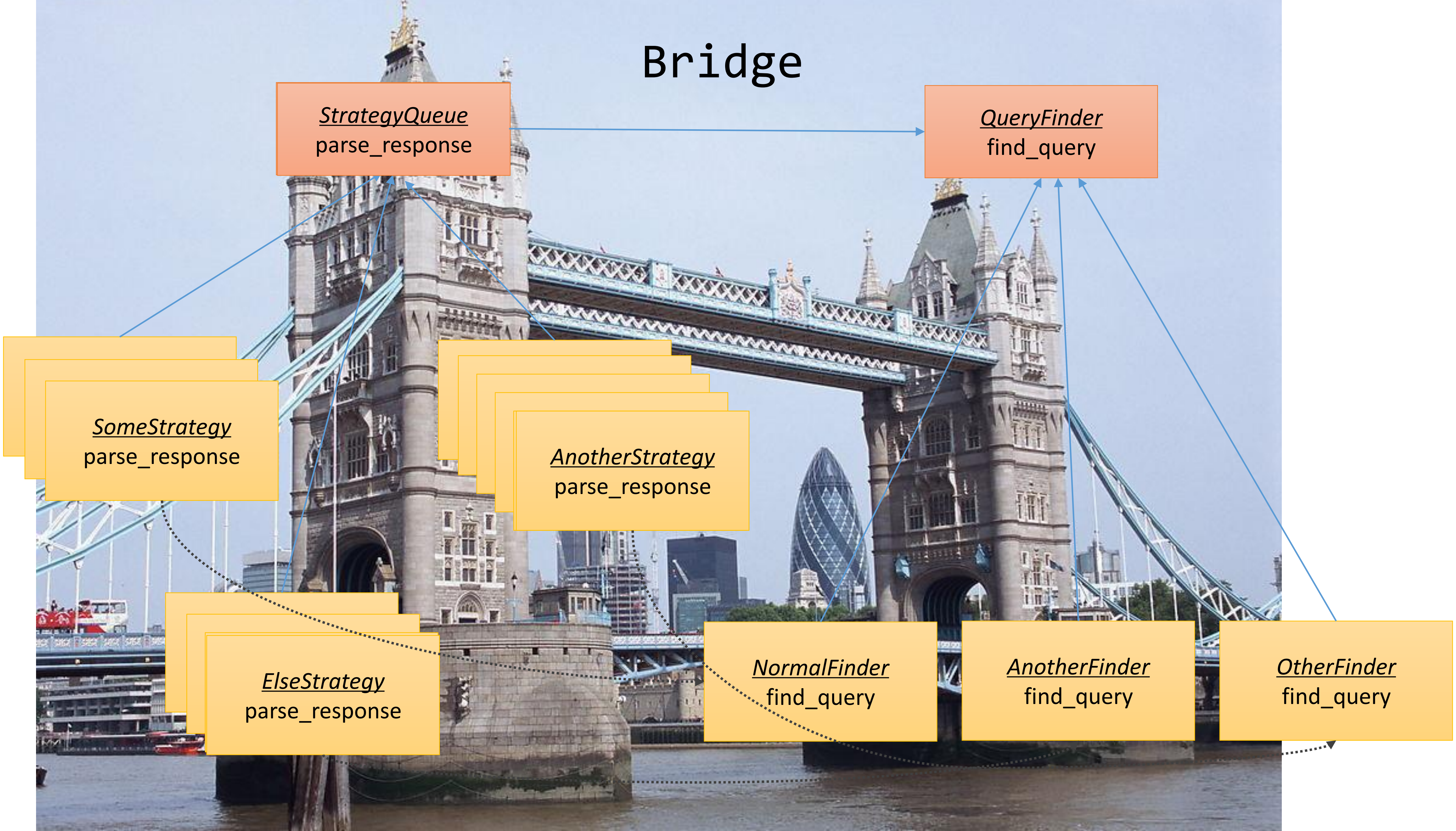


```
QC_ERR_CODE parse_[redacted]_response([redacted]i
        da_interface::analyzed_que
        unsigned int* seg_version)
   Strategy[redacted]Queue::parse_[redacted]

    ///对HIGHRISK检索类型进行调整
    if (normal->has_high[redacted]
            && (normal->highris[redacted]
        _search_type_id = Search_N
        ul_writelog(UL_LOG_DEBUG
```

```
        [redacted]ata* [redacted]Data;
        GET_DATA([redacted]ata, [redacted]Data);

        if [redacted]Data != NULL) {
        [redacted]Data->official_[redacted]
                    [redacted]Data
            ///非高 ///得[redacted]
            if (normal->highrisk_r
                [redacted]ta->is_highri
                _query_info->valid =
            }
```

```
int Strategy[redacted]Queue::parse_[redacted]_response([redacted]ResObj* [redacted]_response_object) {
    CHECK_NULL(-1, [redacted]response_object);
    CHECK_NULL(-1, [redacted]response_object->query_list);
    [redacted]_interface::[redacted]sult* [redacted]normal = [redacted]response_object->[redacted]ResultNormal;
    [redacted]interface::[redacted]esult* [redacted]dual = [redacted]response_object->[redacted]ResultDual;

    unsigned int segment_version[2] = {0, 0};
    extract_segment_version([redacted]dual, segment_version);

    size_t query_index = _finder->find_query_to_be_parsed(*[redacted]normal, *this);
    EXPECT_LT_OR_RETURN(query_index, [redacted]normal->queries_size(), QC_ERR_OK);

    parse_[redacted]_response_common_part([redacted]normal->m_queries(query_index),
        get_[redacted]dual([redacted]dual, query_index), segment_version);

    parse_[redacted]_response_private_part([redacted]normal->m_queries(query_index),
        get_[redacted]dual([redacted]dual, query_index), segment_version);

    return [redacted];
}
```

Bridge


StrategyQueue
parse_response
QueryFinder
find_query
SomeStrategy
parse_response
AnotherStrategy
parse_response
ElseStrategy
parse_response
NormalFinder
find_query
AnotherFinder
find_query
OtherFinder
find_query

编码层面

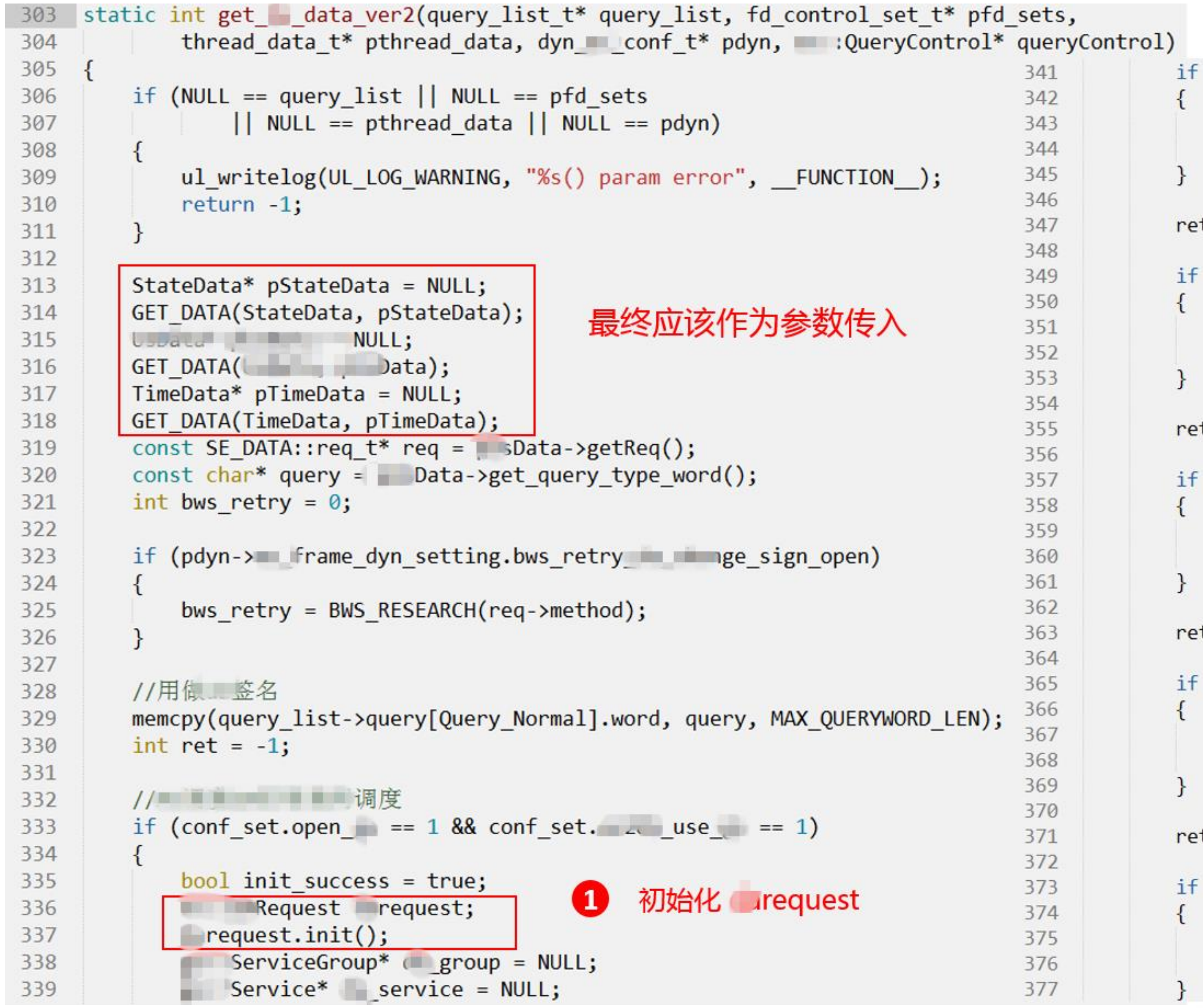
长函数/long method → **抽取方法**/extract method

控制结构重复/duplication → **工具类**/utilities

构造单测数据 → **使用辅助工具**/UT fixture

长函数/long method → **抽取方法**/extract method

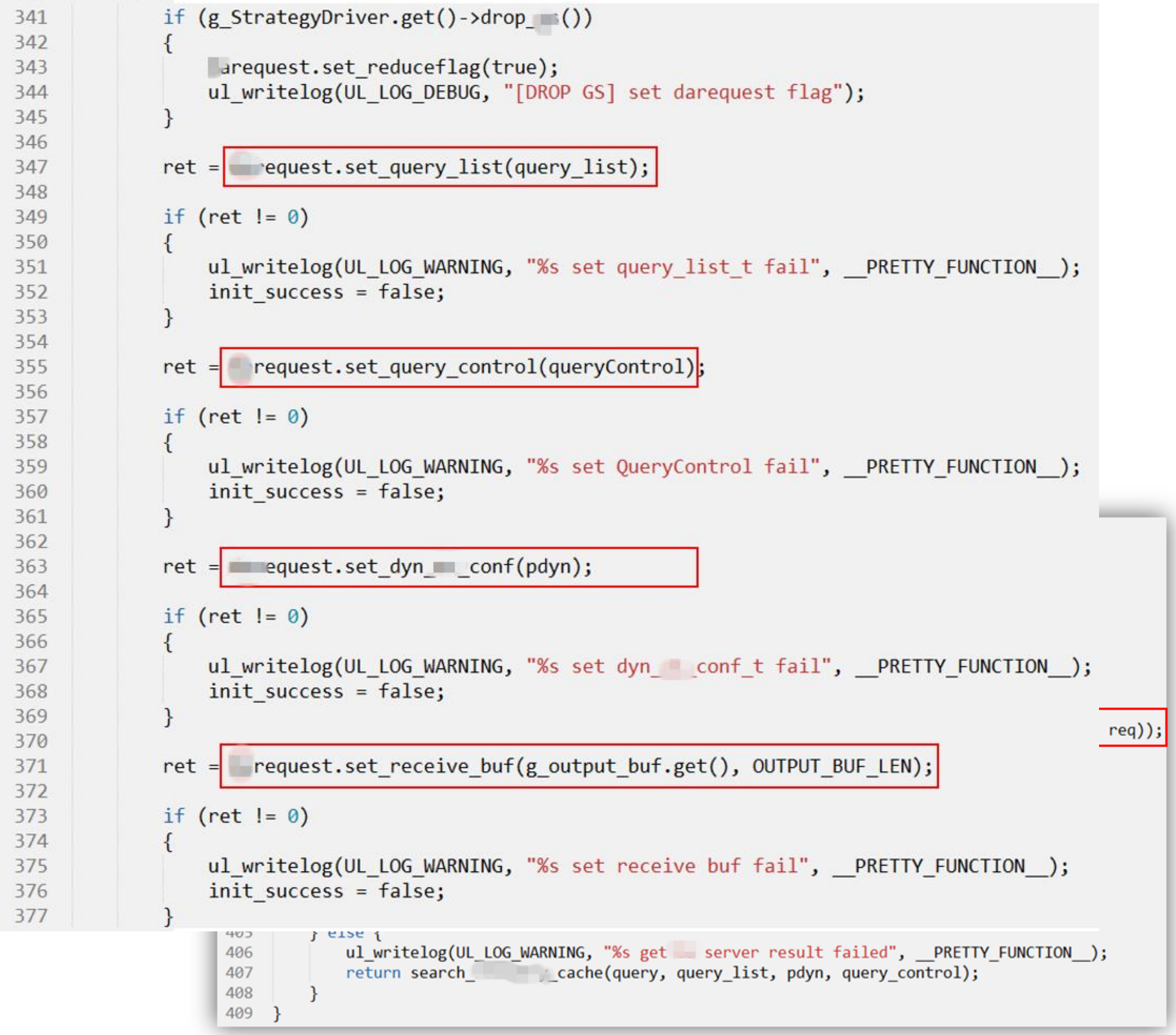
Interaction with Query Analysis – **Before refactoring**



```
static int get_[redacted]_data_ver2(query_list_t* query_list, fd_control_set_t* pfd_sets,
        thread_data_t* pthread_data, dyn_[redacted]_conf_t* pdyn, [redacted]:QueryControl* queryControl)
{
    if (NULL == query_list || NULL == pfd_sets
            || NULL == pthread_data || NULL == pdyn)
    {
        ul_writelog(UL_LOG_WARNING, "%s() param error", __FUNCTION__);
        return -1;
    }

    StateData* pStateData = NULL;
    GET_DATA(StateData, pStateData);
    [redacted] NULL;
    GET_DATA([redacted]Data);
    TimeData* pTimeData = NULL;
    GET_DATA(TimeData, pTimeData);
    const SE_DATA::req_t* req = [redacted]sData->getReq();
    const char* query = [redacted]Data->get_query_type_word();
    int bws_retry = 0;

    if (pdyn->[redacted]frame_dyn_setting.bws_retry[redacted]ge_sign_open)
    {
        bws_retry = BWS_RESEARCH(req->method);
    }

    //用[redacted]签名
    memcpy(query_list->query[Query_Normal].word, query, MAX_QUERYWORD_LEN);
    int ret = -1;

    //[redacted]调度
    if (conf_set.open_[redacted] == 1 && conf_set.[redacted]_use_[redacted] == 1)
    {
        bool init_success = true;
        [redacted]Request [redacted]request;
        [redacted]request.init();
        [redacted]ServiceGroup* [redacted]_group = NULL;
        [redacted]Service* [redacted]_service = NULL;
```

最终应该作为参数传入

1 初始化 [redacted]request

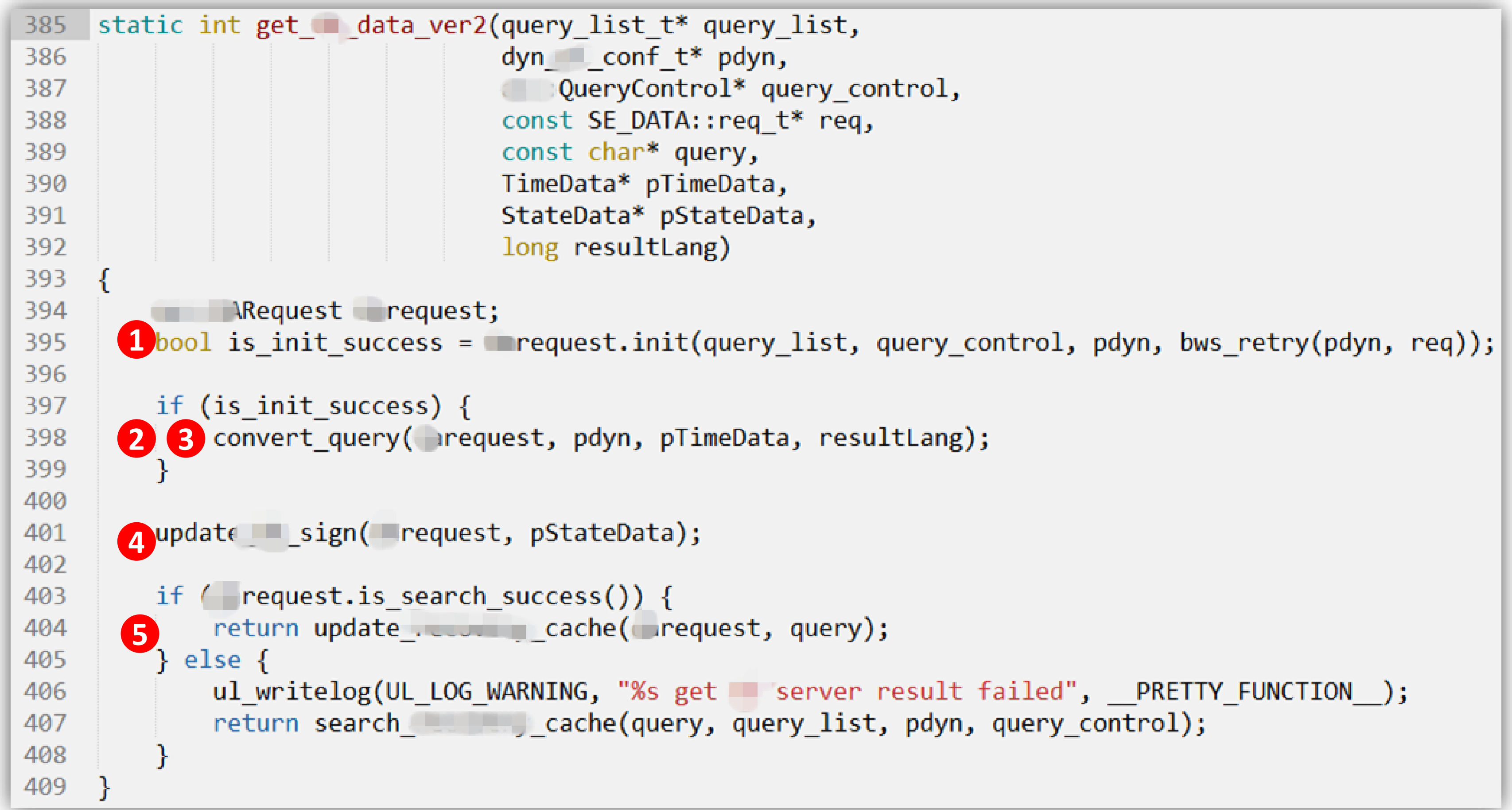


```
        if (g_StrategyDriver.get()->drop_[redacted]())
        {
            [redacted]arequest.set_reduceflag(true);
            ul_writelog(UL_LOG_DEBUG, "[DROP GS] set darequest flag");
        }

        ret = [redacted]request.set_query_list(query_list);

        if (ret != 0)
        {
            ul_writelog(UL_LOG_WARNING, "%s set query_list_t fail", __PRETTY_FUNCTION__);
            init_success = false;
        }

        ret = [redacted]request.set_query_control(queryControl);

        if (ret != 0)
        {
            ul_writelog(UL_LOG_WARNING, "%s set QueryControl fail", __PRETTY_FUNCTION__);
            init_success = false;
        }

        ret = [redacted]request.set_dyn_[redacted]_conf(pdyn);

        if (ret != 0)
        {
            ul_writelog(UL_LOG_WARNING, "%s set dyn_[redacted]conf_t fail", __PRETTY_FUNCTION__);
            init_success = false;
        }

        ret = [redacted]request.set_receive_buf(g_output_buf.get(), OUTPUT_BUF_LEN);

        if (ret != 0)
        {
            ul_writelog(UL_LOG_WARNING, "%s set receive buf fail", __PRETTY_FUNCTION__);
            init_success = false;
        }
```

```
req));
```

```
        } else {
            ul_writelog(UL_LOG_WARNING, "%s get [redacted] server result failed", __PRETTY_FUNCTION__);
            return search_[redacted]cache(query, query_list, pdyn, query_control);
        }
}
```

Interaction with Query Analysis – **After refactoring**



```
static int get_[redacted]_data_ver2(query_list_t* query_list,
                            dyn_[redacted]_conf_t* pdyn,
                            [redacted]QueryControl* query_control,
                            const SE_DATA::req_t* req,
                            const char* query,
                            TimeData* pTimeData,
                            StateData* pStateData,
                            long resultLang)
{
    [redacted]Request [redacted]request;
    bool is_init_success = [redacted]request.init(query_list, query_control, pdyn, bws_retry(pdyn, req));

    if (is_init_success) {
        convert_query([redacted]request, pdyn, pTimeData, resultLang);
    }

    update[redacted]_sign([redacted]request, pStateData);

    if ([redacted]request.is_search_success()) {
        return update_[redacted]_cache([redacted]request, query);
    } else {
        ul_writelog(UL_LOG_WARNING, "%s get [redacted] server result failed", __PRETTY_FUNCTION__);
        return search_[redacted]_cache(query, query_list, pdyn, query_control);
    }
}
```

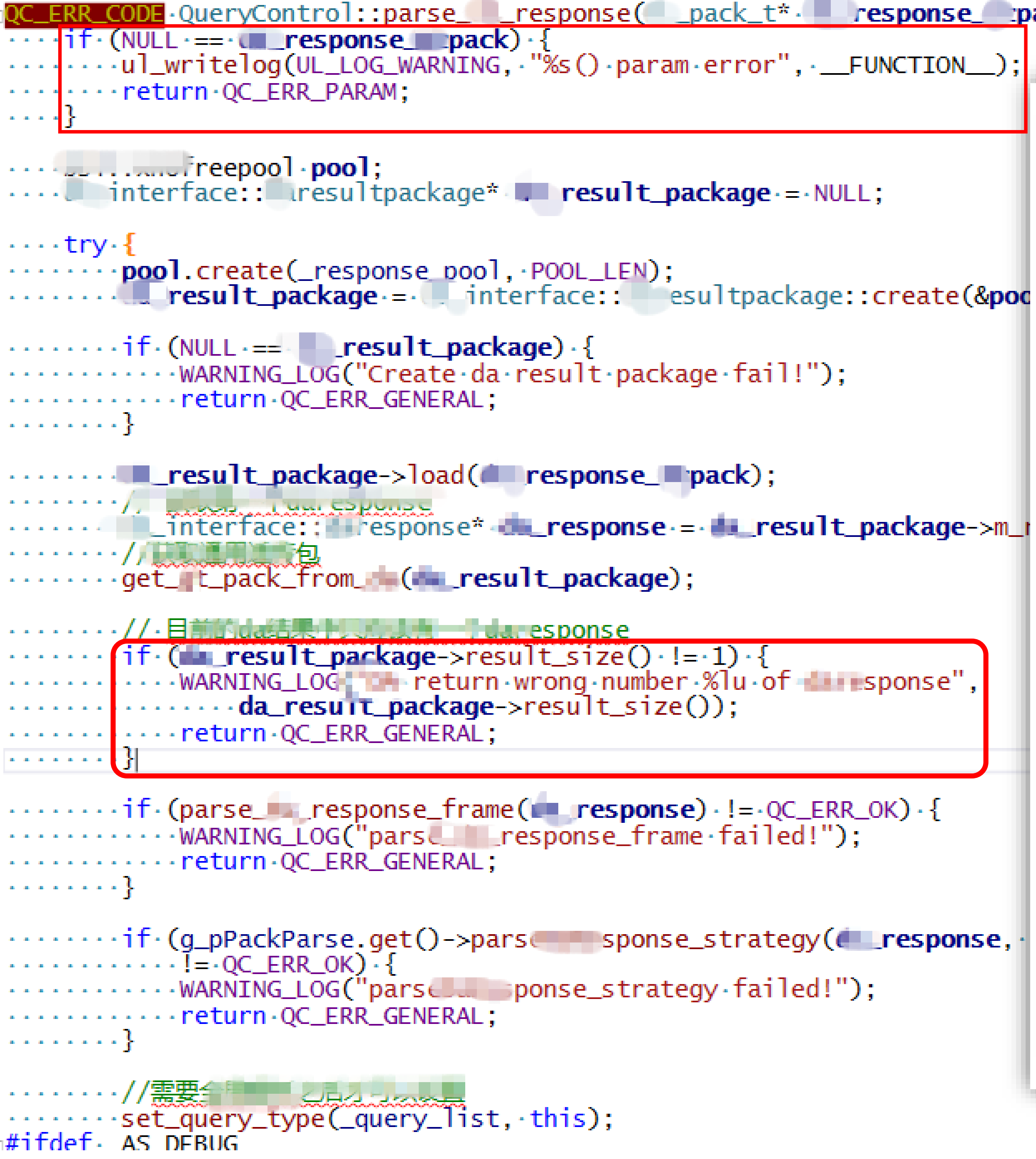
convert_query

1. 初始化 request
2. Get query analysis service
3. 通过Scheduler提交callback
4. 更新签名
5. 更新cache

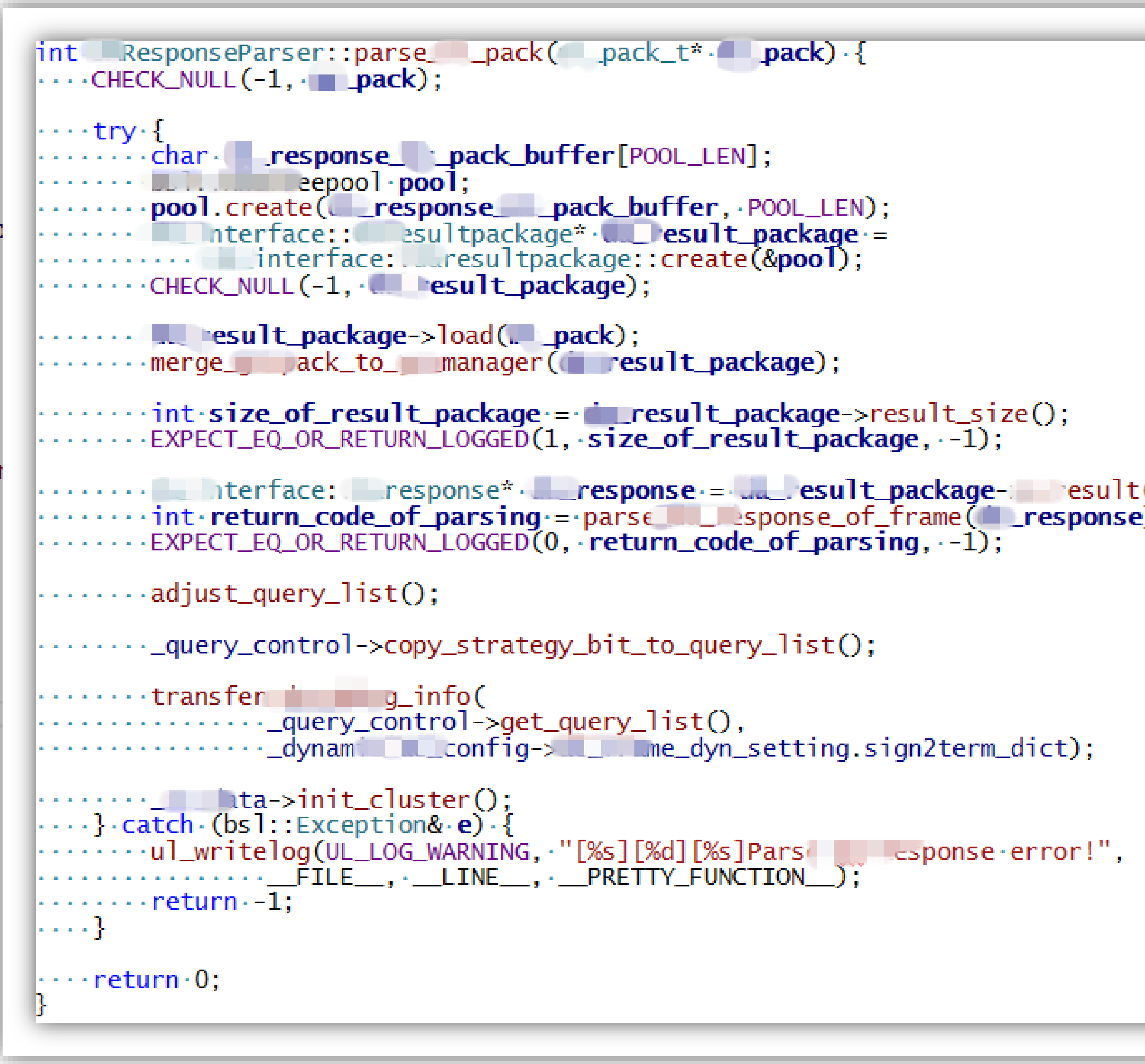
Pattern: Composed Method
均匀一致的抽象层次

Lines: 261 → ~10

控制结构重复/duplication → 工具类/utilities



```
QC_ERR_CODE QueryControl::parse[redacted]_response([redacted]_pack_t* [redacted]response_[redacted]pack) {
    if (NULL == [redacted]_response_[redacted]pack) {
        ul_writelog(UL_LOG_WARNING, "%s() param error", __FUNCTION__);
        return QC_ERR_PARAM;
    }

    [redacted]reepool pool;
    [redacted]interface::[redacted]resultpackage* [redacted] result_package = NULL;

    try {
        pool.create(_response_pool, POOL_LEN);
        [redacted]result_package = [redacted]interface::[redacted]esultpackage::create(&poo

        if (NULL == [redacted]result_package) {
            WARNING_LOG("Create da result package fail!");
            return QC_ERR_GENERAL;
        }

        [redacted]_result_package->load([redacted]response_[redacted]pack);
        //[redacted]
        [redacted]_interface::[redacted]response* [redacted]_response = [redacted]_result_package->m_
        //[redacted]包
        get_[redacted]t_pack_from_[redacted]([redacted]_result_package);

        //目前[redacted]da结果[redacted]一[redacted]da response
        if ([redacted]_result_package->result_size() != 1) {
            WARNING_LOG("[redacted] return wrong number %lu of [redacted]sponse",
                    da_result_package->result_size());
            return QC_ERR_GENERAL;
        }

        if (parse_[redacted]_response_frame([redacted]_response) != QC_ERR_OK) {
            WARNING_LOG("pars[redacted]response_frame failed!");
            return QC_ERR_GENERAL;
        }

        if (g_pPackParse.get()->pars[redacted]sponse_strategy([redacted]_response,
            != QC_ERR_OK) {
            WARNING_LOG("pars[redacted]ponse_strategy failed!");
            return QC_ERR_GENERAL;
        }

        //需要[redacted]之后才可以设置
        set_query_type(_query_list, this);
#ifdef AS_DEBUG
```



```
int [redacted]ResponseParser::parse[redacted]_pack([redacted]_pack_t* [redacted] pack) {
    CHECK_NULL(-1, [redacted]_pack);

    try {
        char [redacted]_response_[redacted]_pack_buffer[POOL_LEN];
        [redacted]eepool pool;
        pool.create([redacted]_response_[redacted]_pack_buffer, POOL_LEN);
        [redacted]nterface::[redacted]esultpackage* [redacted]_result_package =
            [redacted]interface::[redacted]resultpackage::create(&pool);
        CHECK_NULL(-1, [redacted]esult_package);

        [redacted]esult_package->load([redacted]_pack);
        merge_[redacted]ack_to_[redacted]manager([redacted]result_package);

        int size_of_result_package = [redacted]result_package->result_size();
        EXPECT_EQ_OR_RETURN_LOGGED(1, size_of_result_package, -1);

        [redacted]nterface::[redacted]response* [redacted]response = [redacted]esult_package-[redacted]esult
        int return_code_of_parsing = parse[redacted]esponse_of_frame([redacted]_response
        EXPECT_EQ_OR_RETURN_LOGGED(0, return_code_of_parsing, -1);

        adjust_query_list();

        _query_control->copy_strategy_bit_to_query_list();

        transfer[redacted]g_info(
                _query_control->get_query_list(),
                _dynam[redacted]_config->[redacted]me_dyn_setting.sign2term_dict);

        [redacted]ta->init_cluster();
    } catch (bsl::Exception& e) {
        ul_writelog(UL_LOG_WARNING, "[%s][%d][%s]Pars[redacted]sponse error!",
                __FILE__, __LINE__, __PRETTY_FUNCTION__);
        return -1;
    }

    return 0;
}
```

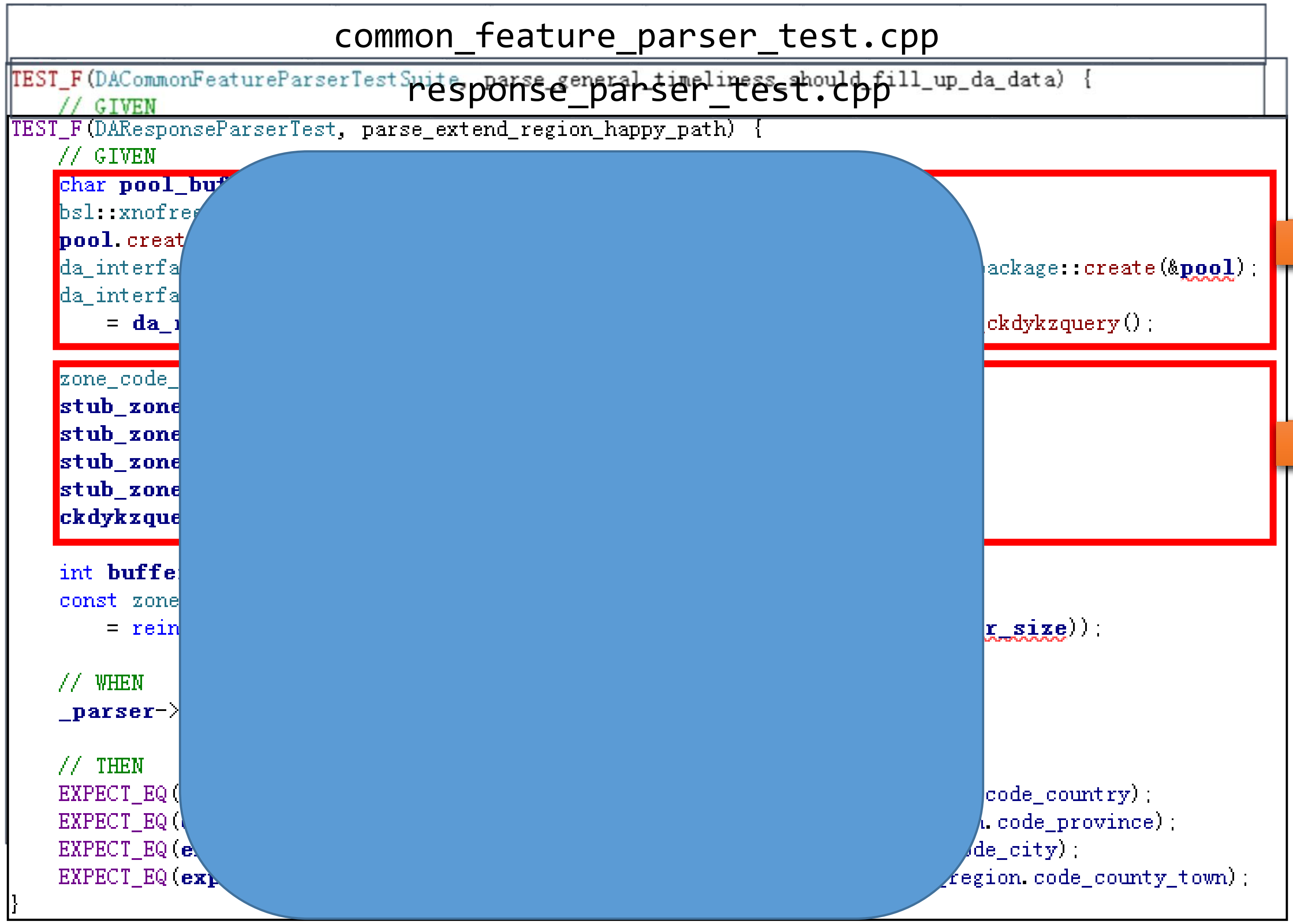
```
Void f() {
     timeval ts, te;
     TimeData* pTimeData = NULL;
     GET_DATA(TimeData, pTimeData);
     gettimeofday(&ts, NULL);
     …… // 处理逻辑
     gettimeofday(&te, NULL);
     pTimeData->parse_da_time += (te.tv_sec -
     ts.tv_sec) * USECS_PER_SEC + (te.tv_usec -
     ts.tv_usec);
}
```

```
Void f() {
  TimeRecorder(&(time_data()->parse_da_time));
  …… // 处理逻辑
}
```

```
XXCallback
-- timeval _parse_start;
-- timeval _parse_end;

在下列函数每一个出错的分支和最后成功的地方都要做
gettimeofday计算时间

XXCallback::on_success() {
     XXCallback::check_lost_and_refuse
     XXCallback::merge
     XXCallback::check_lost
     XXCallback::process_query_type_result
     XXCallback::check_header
     XXCallback::parse_response
     XXCallback::search_more_query_type
}
```

```
XXCallback::on_success() {
 TimeRecorder(&(time_data()->handle_res_bc_time))
 …… // 处理逻辑
}
```

构造单测数据 → 使用辅助工具/UT fixture



common_feature_parser_test.cpp

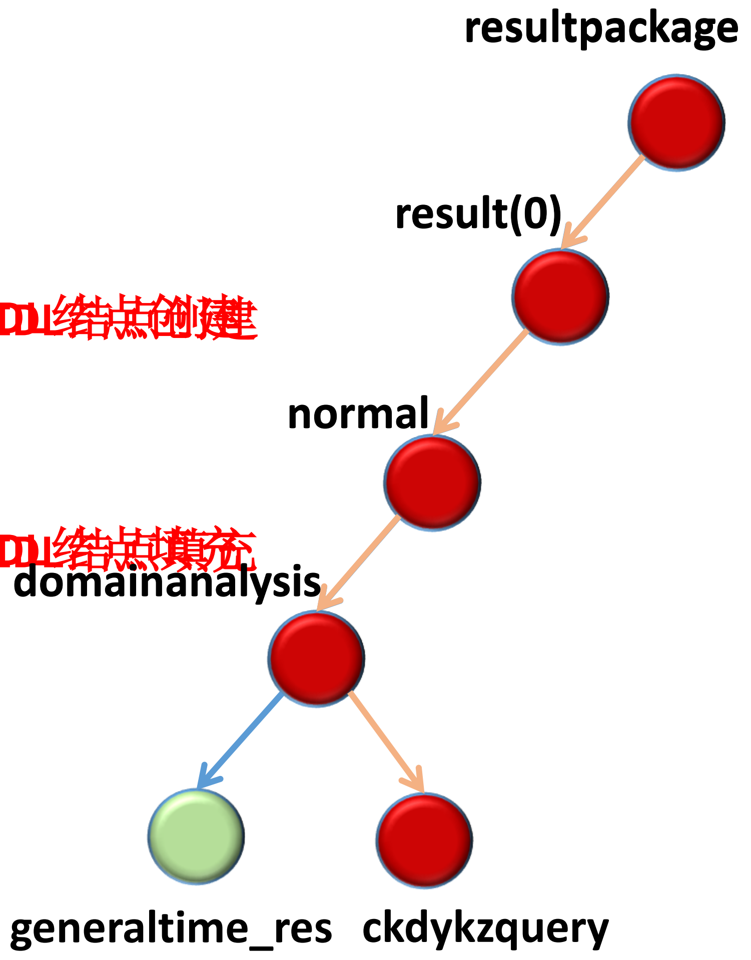
```
TEST_F(DACommonFeatureParserTestSuite, parse_general_timeliness_should_fill_up_da_data) {
    // GIVEN
```

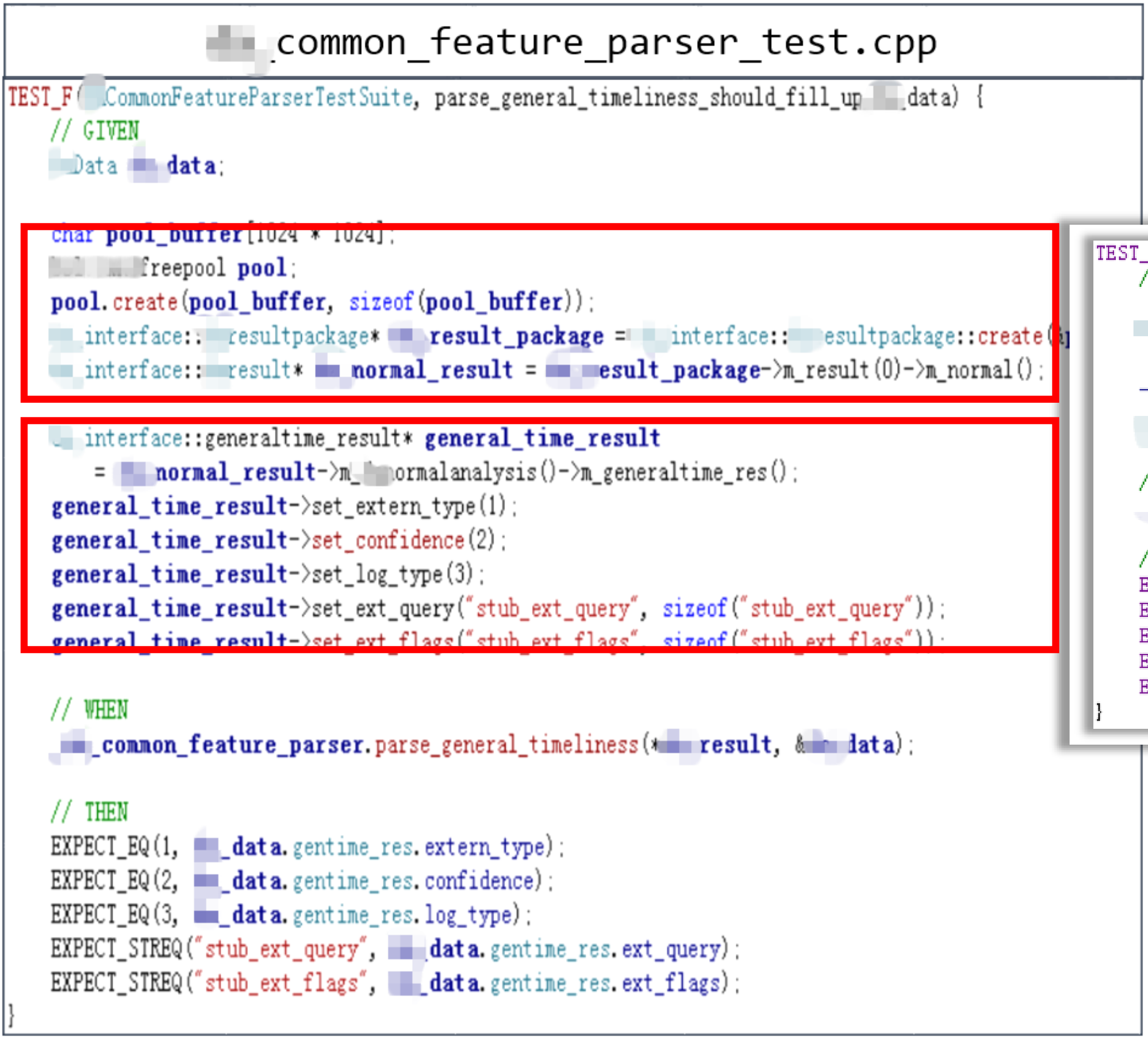
response_parser_test.cpp

```
TEST_F(DAResponseParserTest, parse_extend_region_happy_path) {
    // GIVEN
    char pool_bu
    bsl::xnofree
    pool.creat
    da_interfa                                                    ackage::create(&pool);
    da_interfa
        = da_                                                     ckdykzquery();

    zone_code_
    stub_zone
    stub_zone
    stub_zone
    stub_zone
    ckdykzque

    int buffe
    const zone
        = rein                                                    r_size));

    // WHEN
    _parser->

    // THEN
    EXPECT_EQ(                                                    code_country);
    EXPECT_EQ(                                                    .code_province);
    EXPECT_EQ(e                                                   de_city);
    EXPECT_EQ(exp                                                 region.code_county_town);
}
```

IDL结点创建

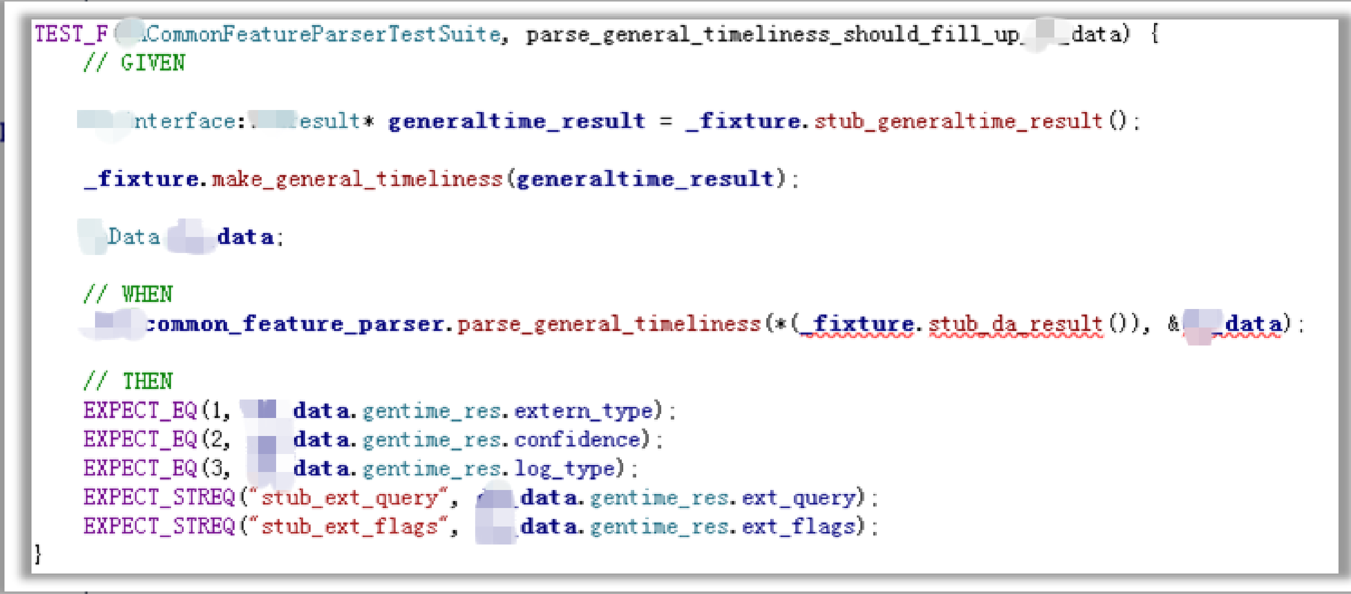
IDL结点填充





[redacted]common_feature_parser_test.cpp

```
TEST_F([redacted]CommonFeatureParserTestSuite, parse_general_timeliness_should_fill_up[redacted]data) {
    // GIVEN
    [redacted]Data [redacted]data;

    char pool_buffer[1024 * 1024];
    [redacted]freepool pool;
    pool.create(pool_buffer, sizeof(pool_buffer));
    [redacted]interface::[redacted]resultpackage* [redacted]result_package = [redacted]interface::[redacted]esultpackage::create(
    [redacted]interface::[redacted]result* [redacted]normal_result = [redacted]esult_package->m_result(0)->m_normal();

    [redacted]interface::generaltime_result* general_time_result
        = [redacted]normal_result->m_[redacted]ormalanalysis()->m_generaltime_res();
    general_time_result->set_extern_type(1);
    general_time_result->set_confidence(2);
    general_time_result->set_log_type(3);
    general_time_result->set_ext_query("stub_ext_query", sizeof("stub_ext_query"));
    general_time_result->set_ext_flags("stub_ext_flags", sizeof("stub_ext_flags"));

    // WHEN
    [redacted]common_feature_parser.parse_general_timeliness(*[redacted]result, &[redacted]data);

    // THEN
    EXPECT_EQ(1, [redacted]data.gentime_res.extern_type);
    EXPECT_EQ(2, [redacted]data.gentime_res.confidence);
    EXPECT_EQ(3, [redacted]data.gentime_res.log_type);
    EXPECT_STREQ("stub_ext_query", [redacted]data.gentime_res.ext_query);
    EXPECT_STREQ("stub_ext_flags", [redacted]data.gentime_res.ext_flags);
}
```



```
TEST_F([redacted]CommonFeatureParserTestSuite, parse_general_timeliness_should_fill_up[redacted]data) {
    // GIVEN

    [redacted]nterface:[redacted]esult* generaltime_result = _fixture.stub_generaltime_result();

    _fixture.make_general_timeliness(generaltime_result);

    [redacted]Data [redacted]data;

    // WHEN
    [redacted]common_feature_parser.parse_general_timeliness(*(_fixture.stub_da_result()), &[redacted]data);

    // THEN
    EXPECT_EQ(1, [redacted]data.gentime_res.extern_type);
    EXPECT_EQ(2, [redacted]data.gentime_res.confidence);
    EXPECT_EQ(3, [redacted]data.gentime_res.log_type);
    EXPECT_STREQ("stub_ext_query", [redacted]data.gentime_res.ext_query);
    EXPECT_STREQ("stub_ext_flags", [redacted]data.gentime_res.ext_flags);
}
```

工具层面

现有工具优化 加快UT编译速度、定制Localbuild

增加质量监控 统计代码复杂度、重复度

使用IDE 代码搬移、重命名、抽取方法自动完成

Thanks

Q & A