体系结构作业 3

171250526 赖童鑫
171250552 刘鹏程
171250558 杨日东
171250575 顾韬

Requirements

- 这一部分是 stakeholders 提出的需求，以及他们对需求重要性的初步评估
- **具体的 scenrios 将会在迭代的过程中写出**

1. functional requirements

	requirements	importance
1	service negotiations	H
2	account management	M
3	trouble call management	M
4	support for multiple concurrent tasks	H
5	validation at near-real-time • of availability of requested service • completion of activities and tasks • integrity of customer data • integrity of the final requested configuration.	H
6	advice on available products and product bundles	M
7	ability to inform the customer about future changes of rates and/or services	M
8	resolution of conflicting events	H
9	support for interrupted and long-lasting conversations	H
10	managing time and date effectively	M
11	integrated support for completing activities	M

2. design constraints

	constraints	importance
1	C4 runs on the cluster of PC and HP9000	H
2	No persistent data caching on the agent workstations	M
3	No DBs at office locations	M

4	High availability can't be achieved by utilizing fault-tolerant hardware	H
5	Support at least 100 service representatives concurrently	H

3. quality attribute requirements

	quality attribute requirements	importance
1*	interoperability: The C4 (Call Center Customer Care System) system interacts with a number of other systems	M
2*	relability: C4 must support for long-lasting conversations	M
3	timeliness(时效性): C4 is an OLTP system,validation of a requested service configuration should be done at near-real-time	H
4*	security: C4 must verify availability of requested service *攻击者模仿一位用户，越权修改一些内部数据。所以要有服务的验证	M
5	scalability: Architect something that is economically viable with a small set of customers and yet can grow to a very large network (i.e. 15 million customers)	H
6	availability: Near 7X24 application availability 全天候服务可用	M
7*	performance: validation of a requested service configuration should be done at near-real-time support a large number (e.g. 400+) of service representatives concurrently	H

Iteration 1

Step 1: Confirm There is Sufficient Requirements

- requirements about business and mission have been prioritized by stakeholders
- constraints have been prioritized
- quality attribute requirements are sufficient
- 下面是各个 quality attribute requirement 的 scenarios

1、scenario 1: 并发服务

Element	Statement
Stimulus	来自同一客户的一系列并发任务
Stimulus Source	某个客户在一次 Business Event 中办理的服务涉及两个及以上数量的任务，或客户要办理多个服务
Environment	一个客服代表正在使用 C4 为一个客户处理 Business Event
Artifact	C4 系统
Response	所有并发任务在本次 Business Event 结束前被正确处理
Response measure	处理 n 个并发任务的时间不超过单独处理每个任务所需的时间总和的 120%（假设），且所有任务在本次 Business Event 结束前都被正确地、没有冲突地处理

2、scenario 2: 冲突解决

Element	Statement
Stimulus	出现事件的冲突
Stimulus Source	不同的 author 就同一个服务发起 Business Event
Environment	不同的 author 通过不同的事件来源，或者通过同一个事件来源与不同的客服代表**同时**就同一个服务进行操作
Artifact	C4 系统和 NOSS 系统
Response	在 C4 向 NOSS 发送服务请求前，所有相关事件都已合并或冲突已解决
Response measure	在不超过正常服务时间的 120%内，冲突被自动合并或解决，或者所有冲突都能够呈现给客服代表以告知客户；到达 NOSS 的所有服务请求都是没有冲突的（假设）

3、scenario 3: timeliness: real-time response

Element	Statement
Stimulus	来自客户提交的服务请求
Stimulus Source	客户提交的服务请求需要被及时反馈
Environment	客服代表使用 C4 为一个客户处理业务
Artifact	C4 系统
Response	客户的请求服务结果需要在限定 x 秒时间内被解决且给出发聘提醒用户服务已完成或无法完成
Response measure	高运行速度的计处理机器和高速存储设备 从输入服务开始计时 预处理输入是否标准，预算服务处理时间，若不标准或者处理

	时间超过 5 秒则反馈重新输入； 若通过预处理和预算进入真正处理阶段，处理结果在 5 秒内未完成，则反馈服务失败，否则将运行结果反馈。

4、scenario 4: 支持中断的持久会话

Element	Statement
Stimulus	客户的对话被客户或代表中断(例如，由于技术原因)、暂停
Stimulus Source	由于技术、环境原因的被动对话中断，以及客户或客服代表的主动中断或暂停
Environment	正常的 Business Event 处理过程，该会话从未中断，或者在经历过中断后都已成功恢复
Artifact	C4 系统
Response	会话信息被完全和正确地保存，并能在会话恢复时重新调出
Response measure	所有会话暂停或中断后，会话上下文在 1 秒内保存，并能够持续保存超过 1 周并能够被完整调出，在重新调出后，会话上下文保持 100%不变（假设）

5、scenario 5: scalability: customer growth

Element	Statement
Stimulus	用户数量的（快速）增长导致瓶颈出现
Stimulus Source	1. 总用户数量持续增长到 15m 2. 新用户数量在一天之内增长 1k
Environment	C4 系统运行在低成本的设备上
Artifact	C4 系统的软硬件
Response	瓶颈能够被快速识别、监控和解决
Response measure	系统能够通过增加资源容量来解决瓶颈，但成本/容量函数必须控制在? 处理瓶颈能够在出现后的 1 分钟（假设）内被发现、解决 当新用户数量在一天之内增长不多于 1k 时，系统能够在当天及其后两天的所有时间内无瓶颈地、100%正常运行 系统能够支持不断增长（到至少 15m）的用户数量

6、scenario 6: availability: All Time available

Element	Statement
Stimulus	系统受不稳定因素影响崩溃
Stimulus Source	系统硬件的不可靠

Environment	C4 系统运行在低成本设备上
Artifact	C4 系统的软件与硬件
Response	硬件上的错误能被迅速识别、监控、恢复解决
Response measure	设计不同节点的备份，统一受一个管理节点管理，当单一节点出现错误时，管理节点调用其他备份节点，并恢复错误节点，管理节点出错时，重新选取管理节点。

Step 2: Choose:an element of the system to decompose

- 我们需要拆解的只有 C4 系统本身

Step 3: Identify the ASRs for the chosen element

#	Architectural Drivers	Importance	Difficulty
1	Requirement 4 Support for mutliple concurrent tasks	H	H
2	Requirement 8 Resolution of conflicting events	H	H
3	Requirement 9 Support for interrupted and long-lasting conversations	H	M
4	Design Constraints 4 High availability cannot be achieved by utilizing fault-tolerant hardware	H	H
5	Scenario 3: timeliness: real-time response	H	H
6	Scenario 5: scalability: customer growth	H	H
7	Requirement 1 Service Negotiations	H	M
8	Requirement 5 Validation at near-real-time	H	M
9	Requirement 2 Account Management	M	M
10	Requirement 3 Trouble Call management	M	M
11	Design Constraints 5	H	M

	Support at least 100 service representatives concurrently		
12	Requirement 10 Managing time and date effectively	M	M
13	Requirement 11 Integrated support for completing activities	M	M
14	Design Constraints 2 No persistent data caching on the agent workstations	M	M
15	Scenario 7: availability: all time available	M	M
16	Requirement 6 Advice on available products and product bundles	M	L
17	Requirement 7 Inform the customers about future changes	M	L
18	Design Constraints 1 C4 runs on the cluster of PC and HP9000	H	L
19	Design Constraints 3 No DBs at office locations	M	L

- 综合考虑 stakeholders 的 ranking，以及 requirements 对最后体系结构的影响，我们选择了以下五个 high-priority requirements 作为 candidate architectural drivers
  - Requirement 4 : Support for interrupted and long-lasting conversations
  - Requirement 8 : Resolution of conflicting events
  - Design Constraints 4 : High availability cannot be achieved by utilizing fault-tolerant hardware
  - Scenario 3 : timeliness : real-time response
  - Scenario 5 : scalability : customer growth
- 其他 ASR 由于对 stakeholders 重要性偏低，或者由于粒度过细等原因对第一次迭代的最终的体系结构影响较小而被删除

Step 4: Choose a design concept that satisfies the ASRs

Step 4.1: Identify design concerns

Design Concerns	Subordinate Concerns
conflicting events resolution	consistency Merge

availability without fault-tolerant hardware	fault recovery
real-time response	response strategies
scalability	hardware support software support

Step 4.2 : List alternative patterns/tactics for subordinate concerns

- Alternative Consistency Tactics

#	Pattern Name	Predictability	Efficiency	Load
1	Lock	high	middle	low
2	Queue	low	high	middle
3	Fixed Time Unreachable	low	middle	low

- **Discriminating parameters**

1. the predictability of the final state of events
2. the efficiency of multiple requests processing
3. the system load of handling multiple requests processing

- **Reasoning**: the predictability should be high to avoid fault, the efficiency should be high to speed up the processing part and load should be low
- **Decision**: Lock
- **Implications**: every time a client request for certain resource, this resource should be locked and nobody else is able to modify the resource content, until the client unlock the resource

- Alternative Merge Tactics

#	Pattern Name	冲突留存时间	冲突解决成功率
1	自动 Merge，失败时 abort 并通知冲突方	取决于自动 Merge 成功率，通常较短	通常较高
2	从不 merge，直接 abort 并通知冲突方	最短	无
3	直接交给客服手动 Merge	较长	最高

- **Discriminating parameters**

1. 冲突在系统中留存的时间
2. 冲突解决的成功率

- **Reasoning**：考虑到系统实时性的要求，手动 Merge 的方法可能无法满足；如果不采取自动 merge 的方法，双方 retry 的次数和时间将取决于双方沟通质量，这难以保证。因此，尽管自动 Merge 存在不确定性，也选择该方法。
- **Decision**：自动 Merge，失败时 abort 并通知冲突方。
- **Implications**：采用某种比较算法进行自动 Merge 尝试，当自动 Merge 返回失败的结果或自动 Merge 的时间超过 2s，则直接 abort 并通知冲突双方存在冲突。

- Alternative fault recovery Conversation Tactics

#	Pattern Name	recovery time	storage cost	communication line loading
1	heartbeat with backup	fast	middle	4 message
2	ping/echo with backup	fast	middle	8 message
3	check point with backup	slow	large	0 message

- **Discriminating parameters**

1. the system recovery time after meeting an error
2. the storage that the backup of data cost
3. the communication line loading to tranfer message

- **Reasoning**: we want the system has high avalibility so the recovery time need to be short.
- heartbeat and ping/echo 在 line loading 方面略有不同，但是 heartbeat 能更好地实时检查部件状态，recovery 的能力更好。check point 需要额外空间去存储信息，并且 recovery 的用时比较长，所以不选择 check point.
- **Decision**: heartbeat/backup
- **Implications**: the DB should support redundancy and have set a manager node to manager/monitoring the data node.
- Alternative response strategies Tactics

#	Pattern Name	response time	efficiency
1	interrupt	short	high
2	polling	long	middle
3	round robin	middle	low

- **Discriminating parameters**

1. the response time between the time the message arrives and the time the system starts to handle the message.
2. the efficiency the system handles the message

- **Reasoning**
  - we hope the system can response to high priority tasks sooner, and process the message with more efficiency. 使用 interrupt 可以及时高效地处理紧急的任务，符合条件
  - polling pattern 中，循环轮询消耗太多时间，而 roubd robin 中时间分片，导致紧急的任务不能及时完成，效率相对低
- **Decision**: interrupt
- **Implications**: when multiple messages or requests arrive, the interrupt pattern will be able to handle tasks with higher importance. Meanwhile, the program effciency improved as the interrupt pattern could make full use of the

processors.

- Alternative Hardware Support(Scalability) Tactics

#	Pattern Name	cost	performance
1	server load balancing	middle	high
2	hardware partitioning	high	high
3	hardware replica	high	middle

- **Discriminating parameters**

1. 需要的花费
2. 最终的性能

- **Reasoning**: 该系统需要对快速增多的服务进行高效的存储与反馈，考虑到 constraints 中有对花费开销的限制，我们需要舍去有高额硬件开销的方式。我们假设存储系统已经有一定数目的存储中心，因此我们采用 server load balancing 的方式，尽可能地保证存储均衡，舍去需要大量冗余存储的方式
- **Decision**: server load balancing
- **Implications**: 每当有新的存储请求时，服务器会主动查找负载较低的存储服务器，将数据存放到其中，并修改相应元数据表便于查找。如果有可靠性需要，可以在负载均衡的基础上做冗余备份，但会有硬件存储的消耗

- Alternative Software Support(Scalability) Tactics

#	Pattern Name	complexity	maintainability	sustainability	coupling
1	third-party tools	low	middle	unknown	high
2	Object Oriented design	middle	high	high	low
3	Process Oriented design	high	middle	low	high

- **Discriminating parameters**

1. 软件的复杂程度
2. 软件的可维护性
3. 软件的可持续性
4. 软件各个模块的耦合性

- **Reasoning**: 该系统需要有很好的扩展性，能对不同量级的请求服务做出不同的反馈，因此需要对软件结构进行设计以便于未来的维护和新功能模块的加入。依赖第三方插件的话，虽然自己软件设计复杂度降低，但是第三方插件的升级和改变会带来不可预估性。面向结构化编程便于开发中小型软件，但是很难添加新的功能模块。面向对象降低各个模块耦合性，可扩展性可维护性都很强

- **Decision**: Object Oriented design
- **Implications**: 每当请求规模到达新的量级，或者产生新的需求时，我们需要对系统的部分模块进行修改或增加新的模块。使用 OO 时我们可以封装软件变化较少的部分，然后对变化的部分进行额外的设计，最终实现软件的 scalability

Step 4.3 : Select patterns/tactics from the list

- **根据 Step 4.2，我们最终选择了一下几个 pattern/tactic**

Lock；自动 Merge，失败时 abort 并通知冲突方；heartbeat/backup；interrupt；server load balancing；Object Oriented design

Step 4.4 : Determine relationship between patterns/tactics and ASRs

#	Pattern Types	Pattern selected	Architectural Driver
1	Consistency	Lock	conflicting events resolution
2	Merge	自动 Merge，失败时 abort 并通知冲突方	conflicting events resolution
3	fault recovery Conversation	heartbeat/backup	availability without fault-tolerant hardware
4	response strategies	interrupt	real-time response
5	Hardware Support(Scalability)	server load balancing	Scalability
6	Software Support(Scalability)	Object Oriented design	Scalability

Step 4.5 : Capture preliminary architectural views

- 在 documenting 中给出最终视图

Step 4.6 : Evaluate and resolve inconsistencies

- 经过检查验证，设计结果与 architectural drivers 相符合，没有出现 inconsistencies

Step 5-7: not included in the report

Iteration 2

Step 1: Confirm There is Sufficient Requirements

- This step is not necessary during each iteration. It was done at the beginning of the ADD process

Step 2: Choose:an element of the system to decompose

- The NOSS handler services element is chosen as the system element to decompos

Step 3: Identify the ASRs for the chosen element

#	Architectural Drivers	Importance	Difficulty
1	Requirement 8: Resolution of conflicting events	H	H
2	Requirement 4: Support for mutliple concurrent tasks	H	H
3	Requirement 9: Support for interrupted and long-lasting conversations	H	M
4	Requirement 5: Validation at near-real-time	H	H
5	Design Constraints 5: Support at least 100 service representatives concurrently	H	M

Step4: Choose a design concept that satisfies the ASRs

Step 4.1: Identify design concerns

Design Concerns	Subordinate Concerns
Task transaction	Conflict resolution mechanism Context persistence
scalability	hardware support software support
timeliness	real-time respose

Step 4.2 : List alternative patterns/tactics for subordinate concerns

Alternative Conflict Resolution Mechanism Tactics

#	Pattern Name	Predictability	Efficiency	Load
1	identifier	high	middle	low
2	read-only	low	low	low
3	Monitor	low	high	high

- **Discriminating parameters**

1 the predictability of the final state of events

2 the efficiency of multiple requests processing

3 the system load of handling multiple requests processing

- **Reasoning**: the predictability should be high to avoid fault, the efficiency should be high to speed up the processing part and load should be low
  只读策略无法满足数据修改的需求，灵活性低，对每条服务请求进行标志增加通 讯负担，影响反馈速度
- **Decision**: Monitor
- **Implications**: 每次监听到一个冲突时，冲突双方停止传输，双方在规定的不同时间间隔后再次传输服务请求。

Alternative Context Persistence Tactics

#	Pattern Name	cost	R/W speed
1	store in Context Storage only	lowest	Medium
2	store in cache, then Context Storage	slightly higher than 1	Fast when hit

- **Discriminating parameters**

1. The cost of pattern, measure by money
2. Read and write speed

- **Reasoning**: 直接从持久数据库存取的成本低，但是速度比使用 cache 慢，而且在并发场景中会成为瓶颈；不能直接支持事务的实现
- **Decision**: cache+Context Storage
- **Implications**: 存在 concurrent tasks 事务时，tasks 全部存入 cache，全部成功后从 cache 中退出；所有到来的 task 先在 cache 中驻留，等待冲突敏感期结束后再释放给外部

Alternative Hardware support Tactics

#	Pattern Name	difficulty	cost	Upper limit
1	public cloud platform	easy	high	high
2	self-development	difficult	low	no

- **Discriminating parameters**

1 the difficulty of the tactics

2 the cost of the total process

3 how many servers can we use

- **Reasoning**: 我们希望硬件系统有很高的拓展性，且价格便宜，在之后的自主拓展上有很高灵活性和上限。公有云平台使用起来方便快捷，但受到平台限制，上限不明

确，且当数据量极大时性价比不如自主开发，不具有更好的拓展性。

- **Decision**: self-development
- **Implications**: 自主研发一个高拓展性服务器集群系统

Alternative Software support Tactics

#	Pattern Name	difficulty	Upper limit
1	Event driven architecture	difficult	high
2	Distributed message queue	easy	high

- **Discriminating parameters**

1 the difficulty of the tactics

2 Upper limit of expansion

- **Reasoning**: 软件系统应该支持高拓展性，后期拓展过程容易且上限高。
  分布消息队列模型简单且明确，吞吐量大，延迟低，有容灾能力。
  事件驱动模型比较复杂，在领域事件范围不是太大的情况下效益低。
- **Decision**: Distributed MQ
- **Implicationsre**：使用生产者消费者模型，发送者和接收者可以有多种部署实例，但共用一个队列。

Alternative Real-time Response Tactics

#	Pattern Name	delay	cost	efficency
1	long polling	little	middle	middle
2	polling	little	little	low
3	websocket	little	little	high

- **Discriminating parameters**

1 该策略导致的延迟。

2 策略等待总时间

3 策略获取反馈的效率（获取的准确性和完备性）

- **Reasoning**: 实时反馈应该低延迟，且策略的计算消耗要低，获得的反馈要准确完备，系统要确保用户得到了反馈。
- **Decision**: websocket
- **Implicationsre**：在每个功能模块间使用 TCP/IP 通讯，将通讯数据打包成 socket 在模块间传输。

Step 4.3 : Select patterns/tactics from the list

根据 4.2 我们选择以下 pattern/tactic

Moniter，cache+Context Storage，self-development，Distributed MQ，websocket

Step 4.4 : Determine relationship between patterns/tactics and ASRs

#	Pattern Types	Pattern selected	Architectural Driver
1	Conflict Resolution Mechanism	Monitor	Task transaction
2	Context Persistence	cache+Context Storage	Task transaction
3	Hardware support	self-development	scalability
4	Software support	Distributed MQ	scalability
5	Real-time Response	websocket	timeliness

Step 4.5 : Capture preliminary architectural views

- 在 documenting 中给出最终视图

Step 4.6 : Evaluate and resolve inconsistencies

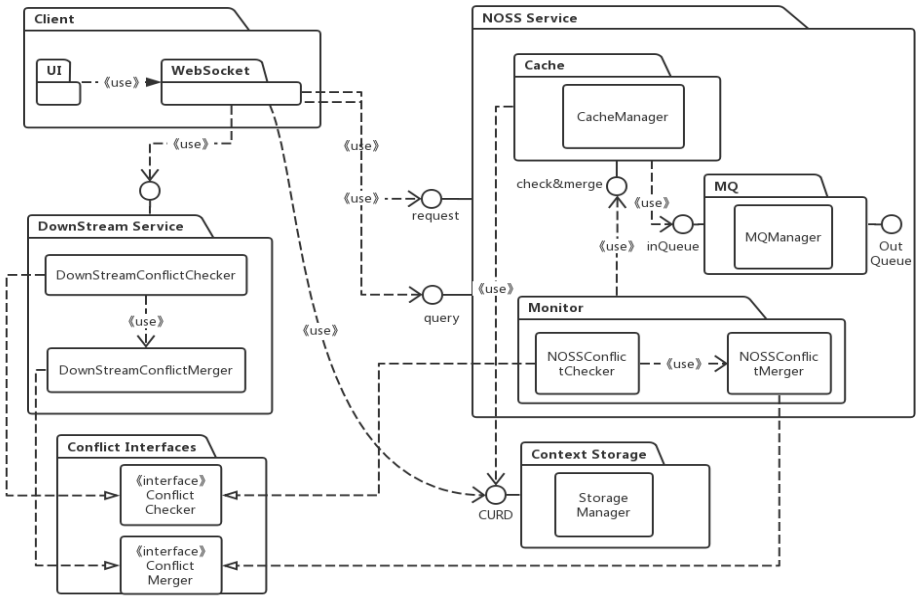
- 经过检查验证，设计结果与 architectural drivers 相符合，没有出现 inconsistencies

Step 5-7: not included in the report

Final Software architecture documentation

1. C4 Module view

Section 1. Primary Presentation



Section 2. Element Catalog

Sec 2.A: Elements

Client=客服客户端模块，Client.UI=客服客户端 UI 模块，Client.WebSocket=客服客户端网络通信模块；ConflictInterfaces=冲突服务接口包；DownStreamService=下游系统服务模块，DownStreamConflictChecker=下游系统模块冲突检测单元，DownStreamConflictMerger=下游系统模块冲突合并单元；NOSSService=NOSS 服务模块；Cache.CacheManager=缓存管理单元；MQ.MQManager=消息队列管理单元；Monitor.NOSSConflictChecker=NOSS 冲突检测单元，

Monitor.NOSSConflictMerger= NOSS 冲突合并单元；

ContextStorage.StorageManager=上下文存储管理单元

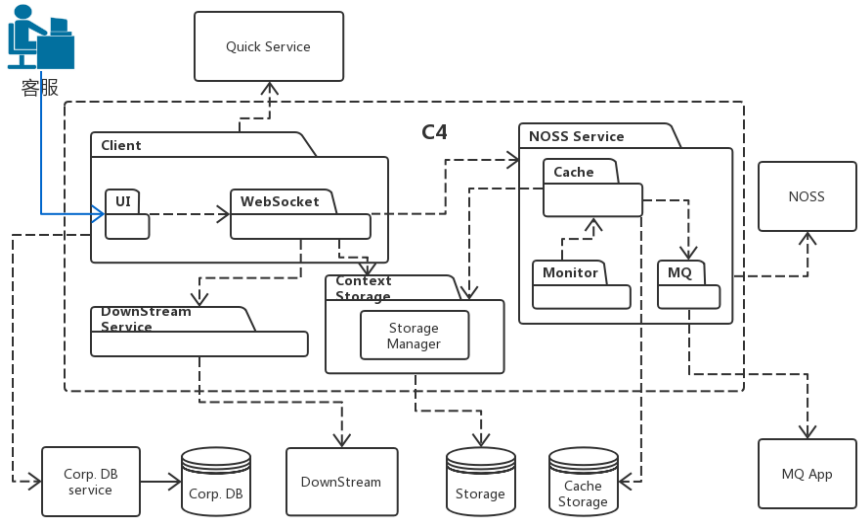
Sec 2.B: Relations

- <<use>>：依赖关系，使用了模块提供的接口，或直接调用模块/单元的方法
- - -|>：实现关系，具体类实现了接口

Sec 2.C: Interfaces

NOSSService 模块向外提供 request 与 query 接口，供客服客户端进行请求和查询操作；MQ 向父模块内提供请求入队接口，向外提供请求出队接口；DownStreamService 模块提供接口，供客服客户端请求服务；ContextStorage 模块向外提供 CURD 功能接口。

Section 3. Context Diagram



Section 4. Variability Guide

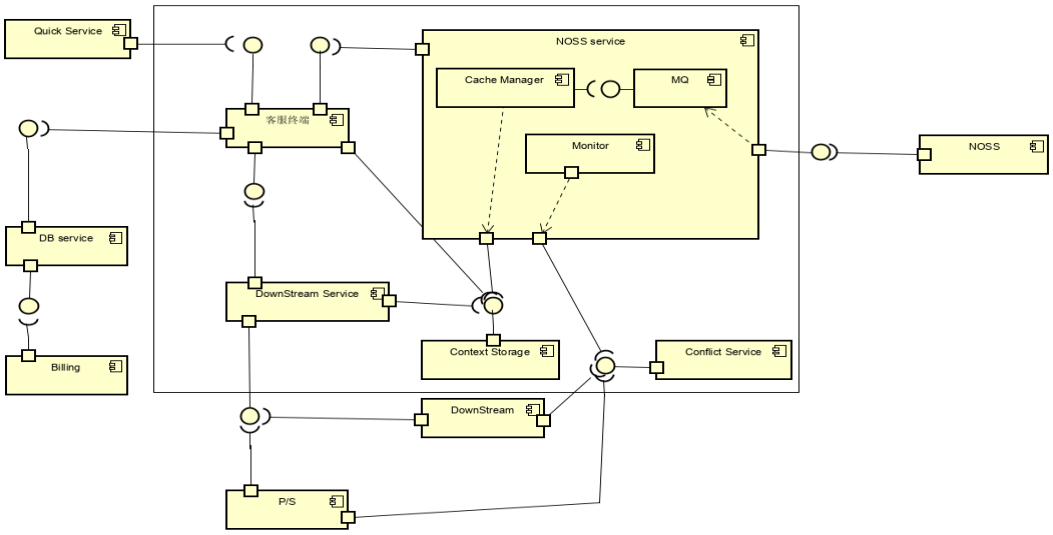
Cache 采用 Redis 或者 Memcached，具体采取哪个需要考察 HP9000 的性能；Context Storage 需要根据业务数据的结构化程度选择，若结构化程度高，则采用 Mysql，若含有难以结构化的部分，则采用文件存储；如果客服使用的 PC 可以使用浏览器，则用户界面选择建立在浏览器中，否则使用窗口 UI

Section 5. Rationale

只读策略无法满足数据修改的需求，灵活性低，对每条服务请求进行标志增加通讯负担，影响反馈速度，所以选择 Monitor；直接从持久数据库存取比使用 cache 慢，而且在并发场景中会成为瓶颈、不能直接支持事务的实现，所以选择 cache

2. **C4 component-connector view**

Section 1. Primary Presentation



Section 2. Element Catalog

Sec 2.A: Elements

Quick Service: 请求快速服务组件；DB Service: 数据存储组件; DownStream: 下游系统组件；DownStream Service: 下游系统服务组件; Context Storage: 上下文存储组件；Conflict Service: 冲突协调组件；Cache Manager: cache 处理模块；MQ：消息队列处理模块; NOSS Service: NOSS 服务模块；Monitor: NOSS 冲突监控模块；

Sec 2.B: Relations

以圆圈为末尾的线段：provided interface；以圆弧为末尾的线段：required interface；虚线加箭头：dependency；直线：connector

Sec 2.C: Interfaces

NOSS Service 提供协调处理 NOSS 信息的接口，Conflict Service 提供处理冲突的接口，实用终端提供处理各项服务的接口，Context Storage 提供 C4 内部上下文存储的接口

Section 3. Context Diagram

如上图，C4 system 和外界用矩形边界隔离开，对每种类型的外部请求都使用对应的 service 来处理

Section 4. Variability Guide

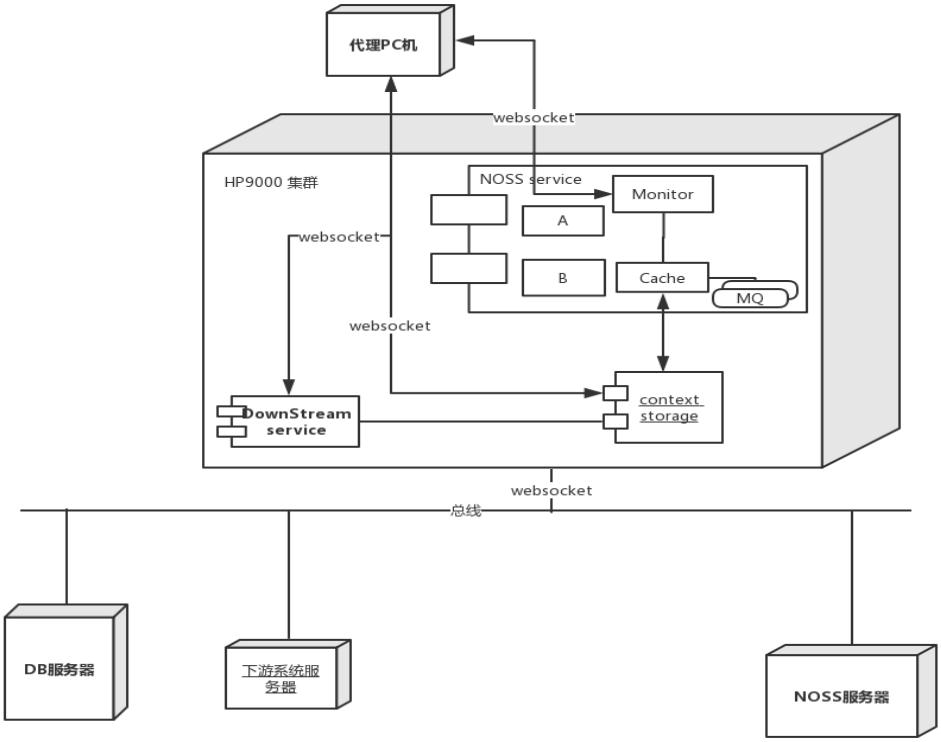
如果有短时间内的请求数量过大的可能，需要在 C4 Context Storage 中做好分布式存储来防止由于单点负载过大而产生崩溃。其他的静态模块 variability 已经在 module view 中提出。

Section 5: Rationale

将各个外部服务接口拆解下来，使得各个 service 对未来外部的变动有更好的适应性，便于对各个接口做扩展服务，也有利于未来的硬件升级，体现了 scalability。使用 context storage 做 C4 内部的缓存，体现了使用的 backup tactic。C4 中使用了专门的 Conflict service 做冲突调度，能很好地应对不同源的冲突请求，同时对新 request 源的加入，也有着良好的扩展性，体现了 consistency 和 scalability

3. C4 Allocation view

Section 1. Primary Presentation



Section 2. Element Catalog

Sec 2.A: Elements

代理 PC：服务请求发出地；DB 服务器： 数据存储服务器; DownStream：下游系统服务器；HP9000 集群：C4 服务运行服务器 ；NOSS Service：C4 处理 NOSS 服务的服务器，有一台备用服务器 B 和一台主服务器 A；Monitor 负责监听冲突 Cache 负责处理中断；DownStream service：负责处理与下游服务器的交互，且处理与 NOSS Service 产生的冲突

Sec 2.B: Relations

直线加箭头表示依赖关系，直线表示共享关系，矩形框表示物理上临近的关系

Sec 2.C: Interfaces

无软件接口，只有物理节点提供了底层服务，如 DB 服务器提供了大容量数据存储服务，cache 提供了临时上下文数据的服务，websocket 提供远距离数据传输的服务

Section 3. Context Diagram

C4 部署的 context 如上图所示

Section 4. Variability Guide

如果有用户数据量快速增长的可能，需要对 server 中做好分布式架构来防止由于单点负载过大而产生崩溃，便于进行热备份。其他的 variability 已经在之前的 view 中提出。

Section 5: Rationale

这里采用集群结构，方便进行负载均衡，体现出软件结构的 scalability。同时 cache+storage 结构体现在部署模型中在 NOSS service 部分使用 cache 缓存，并使用 context storage 作为 C4 内部的集中存储

3.4 C4 cross-view

Section 1. View Catalog

1. Module View

A module is an implementation unit of software that provides a coherent unit of functionality

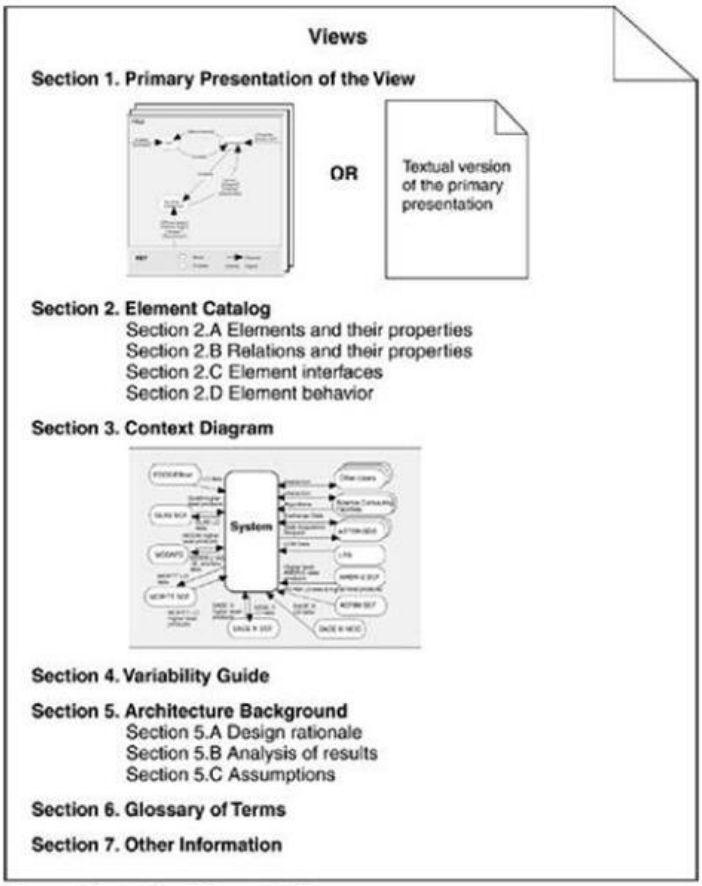
2. Allocation View

Express the allocation of software elements to nonsoftware structs。

3. Component-Connector View

Component-Connector View represent unit of execution plus the pathways and protocols of their interaction。

Section 2. View template

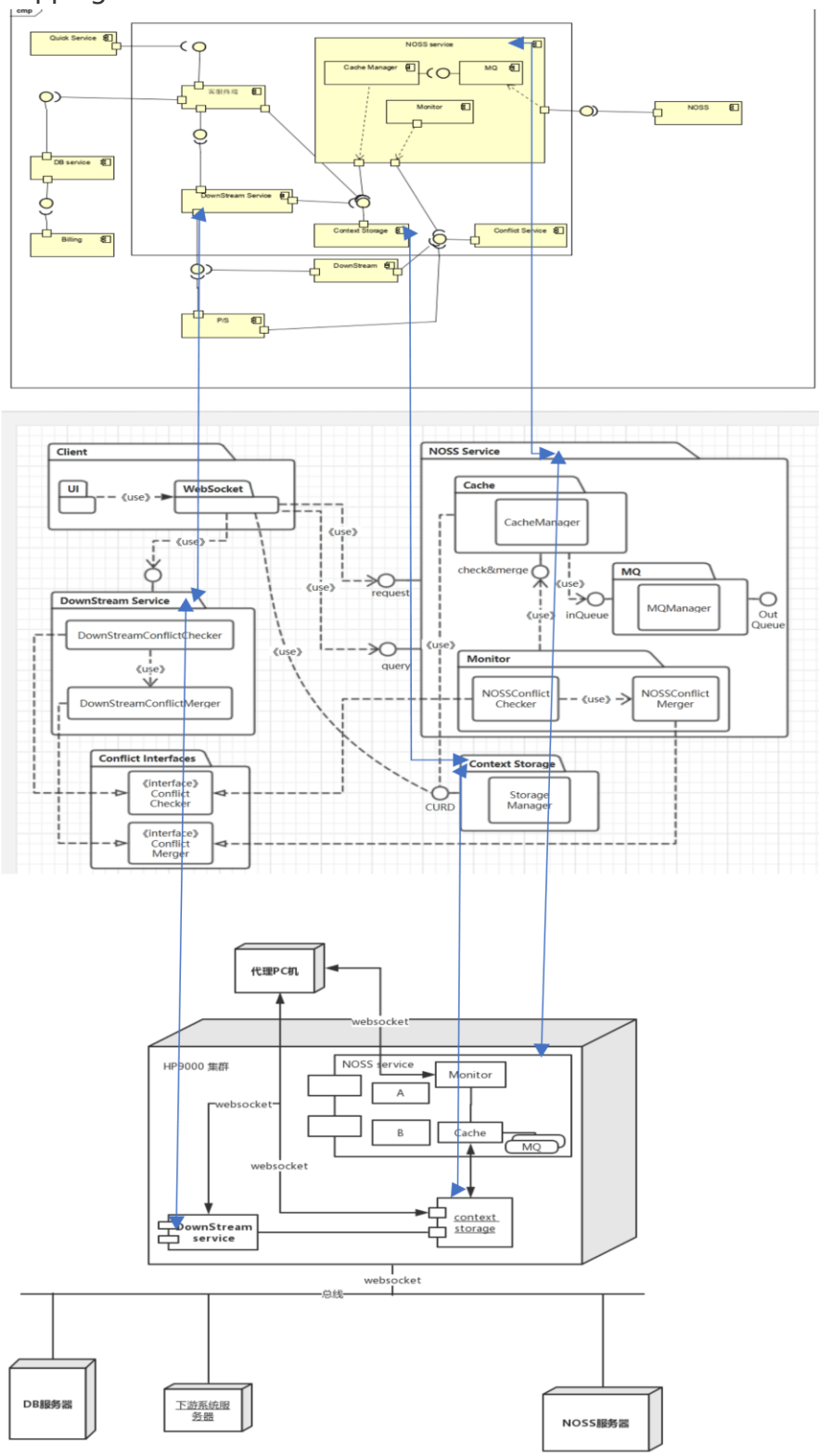


Section 3. System Overview

C4 is an OLTP system that the primary function of the system is to support interactions with the customers that request new services (ex: new phone lines), changes in the configuration of the existing services (ex: phone number changes,

long-distance company changes, or relocation), or report problems.

Section 4.Mapping between Views



Section 5. RATIONALE

三个视角下，都设计了对 NOSS service 的 Monitor、cache、MQ 来实现对对话的即时保存与恢复，模块的分离易于错误的探查与数据验证，体现了 availability 与

consistency，接口的分离体现了可修改性与可扩展性。

Individual remark

顾韬

感触：在使用 ADD 方法中，我感受到需求分析对于后期系统设计的影响是很深的。我对原始的 C4 文档进行需求分析中没能把握好需求的重要性排序，导致后期分析关注点经常有偏移。此外，ADD 方法对系统拆解要注意粒度的把控，过度分解导致单次迭代难度太高，需要考虑问题太多，分解不完全则无法进行总体设计。

工作：整理提炼需求，第二次迭代的 4.2，4.3 部分，以及 allocation view

赖童鑫

感触：通过对 add 的使用，我认识到在设计一个大型系统时，需要对纷繁的需求做出准确提炼，把握对系统构架影响最大的需求集合，然后在此之上开始对系统架构模式进行衡量选择。

工作：需求整理，cross-view，pattern 选取比较。

刘鹏程

感触：在使用 ADD 方法的过程中，我感受到在设计一个复杂系统体系结构的过程中，是如何抽取需求中的重要信息，并将其转化成最终的体系结构设计的。整个过程将繁杂而难以确定的需求信息一步步变得充分明确，为我们解决复杂软件设计的问题提供了清晰的道路。我发现由于自己技术经验不足，因此在确定具体的 patterns/tactic 时，一方面很难针对 design concerns 找到足够的 patterns/tactics 来进行选择，另一方面选择 patterns /tactics 时不确定是否找到合适的决定性因素。这些背景知识需要日后不断学习补足。

工作：参与整个迭代过程的任务和流程梳理，参与 requirements 对体系结构重要性的评审，决定第一阶段的 design concerns，并对各个 subordinate concerns 确定最终使用的 pattern/tactic，以及最终文档里 component-connector view 的编写工作。

杨日东

感触：我使用 ADD 方法的感触是：首先，由于在第一次迭代时，我们所面对的是一个完全未开发过的大型系统，因此使用 ADD 方法进行第一次迭代会产生大量的需求，带来较大的工作量，这使得第一次迭代的过程让我比较迷茫，但是第二次迭代能够针对某一个 element 进行设计，目标更清晰。其次，确定 design concern 是我在使用 ADD 方法时遇到的最大的难题，我认为 ADD 方法中的步骤 4.2 结果的质量实际上非常取决于步骤 4.1 中对 design concern 的确定，如果 design concern 没有确定好，那么后面选择 patterns/tactics 时就会偏离方向。

工作：我在小组中主要参与了第一次迭代的 Scenarios 编写、design concern 之冲突解决机制的解决，以及第二次迭代中 design concern 之 Task transaction 的解决，还有最终文档中 Module View 的编写。