

GENESYS 2024 UAF View Specification Definitions



UAF View Specification Definitions

- Each View Specification that is based on a Block Definition Diagram has a unique set of classes and relations associated with it.
 - Each View Specification has a or set of root entities associated with it.
 - Root entities are called out in the definition diagram with a dashed border
 - Each View Specification has a root entity relation associated with it.
 - Root relationships are called out with an *

Diagrams in this presentation are maps to each Block Definition Diagram in a GENESYS UAF Project

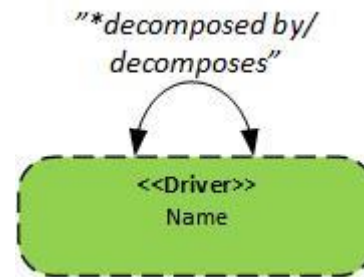
Available View Specification Definitions

UAF	Motivation Mv	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Sequences Sq	Information ^c If	Parameters ^d Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Architecture Management ^a Am	Architecture Principles Am-Mv	Architecture Extensions Am-Tx ^e	Architecture Views Am-Sr	Architecture References Am-Cn	Architecture Development Method Am-Pr	Architecture Status Am-St		Dictionary Am-If	Architecture Parameters Am-Pm	Architecture Constraints Am-Ct	Architecture Roadmap Am-Rm	Architecture Traceability Am-Tr
Summary & Overview Sm-Ov												
Strategic St	Strategic Motivation St-Mv	Strategic Taxonomy St-Tx	Strategic Structure St-Sr	Strategic Connectivity St-Cn	Strategic Processes St-Pr	Strategic States St-St		Strategic Information St-If		Strategic Constraints St-Ct	Strategic Deployment, St-Rm-D Strategic Phasing St-Rm-P	Strategic Traceability St-Tr
Operational Op		Operational Taxonomy Op-Tx	Operational Structure Op-Sr	Operational Connectivity Op-Cn	Operational Processes Op-Pr	Operational States Op-St	Operational Sequences Op-Sq			Operational Constraints Op-Ct		Operational Traceability Op-Tr
Services Sv		Services Taxonomy Sv-Tx	Services Structure Sv-Sr	Services Connectivity Sv-Cn	Services Processes Sv-Pr	Services States Sv-St	Services Sequences Sv-Sq	Operational Information Op-If		Services Constraints Sv-Ct	Services Roadmap Sv-Rm	Services Traceability Sv-Tr
Personnel Ps	Requirements Rq-Mv	Personnel Taxonomy Ps-Tx	Personnel Structure Ps-Sr	Personnel Connectivity Ps-Cn	Personnel Processes Ps-Pr	Personnel States Ps-St	Personnel Sequences Ps-Sq			Personnel Availability Ps-Rm-A Competence, Drivers, Performance Ps-Ct Personnel Evolution PS-Rm-E Personnel Forecast Ps-Rm-F		Personnel Traceability Ps-Tr
Resources Rs		Resources Taxonomy Rs-Tx	Resources Structure Rs-Sr	Resources Connectivity Rs-Cn	Resources Processes Rs-Pr	Resources States Rs-St	Resources Sequences Rs-Sq	Resources Information Rs-If		Resources Constraints Rs-Ct	Resources evolution Rs-Rm-E Resources forecast Rs-Rm-F	Resources Traceability Rs-Tr
Security Sc	Security Controls Sc-Mv	Security Taxonomy Sc-Tx	Security Structure Sc-Sr	Security Connectivity Sc-Cn	Security Processes Sc-Pr					Security Constraints Sc-Ct		Security Traceability Sc-Tr
Projects PJ		Projects Taxonomy PJ-Tx	Projects Structure PJ-Sr	Projects Connectivity PJ-Cn	Projects Processes PJ-Pr						Projects Roadmap PJ-Rm	Projects Traceability PJ-Tr
Standards Sd		Standards Taxonomy Sd-Tx	Standards Structure Sd-Sr								Standards Roadmap Sd-Rm	Standards Traceability Sd-Tr
Actual Resources Ar			Actual Resources Structure, Ar-Sr	Actual Resources Connectivity, Ar-Cn		Simulation ^b				Parametric Execution/Evaluation ^b		

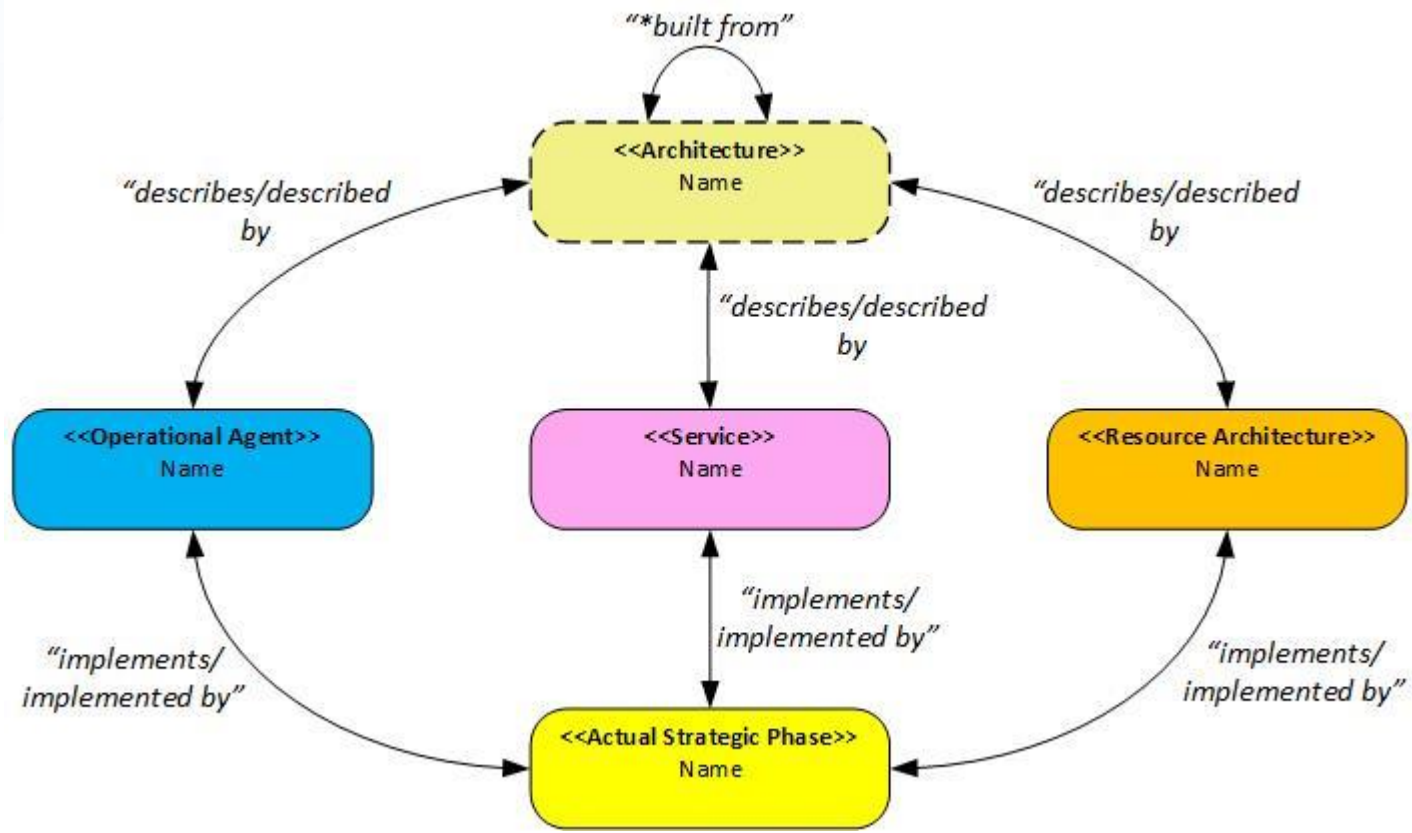
Environment En-Pm-E
and
Measurements Me-Pm-M
and
Risks Rk-Pm-R

Architecture Management (Am)

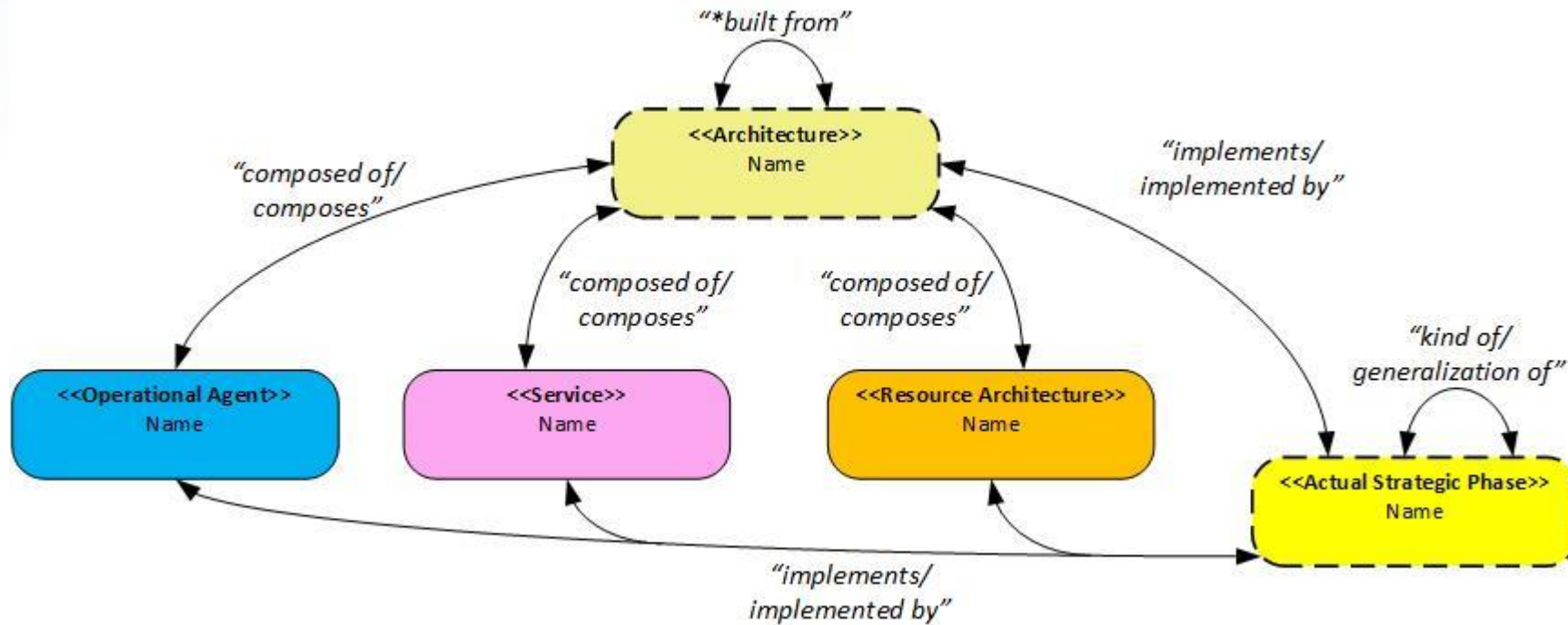
Architecture Principles (Am-Mv)



Architecture Extensions (Am-Tx)

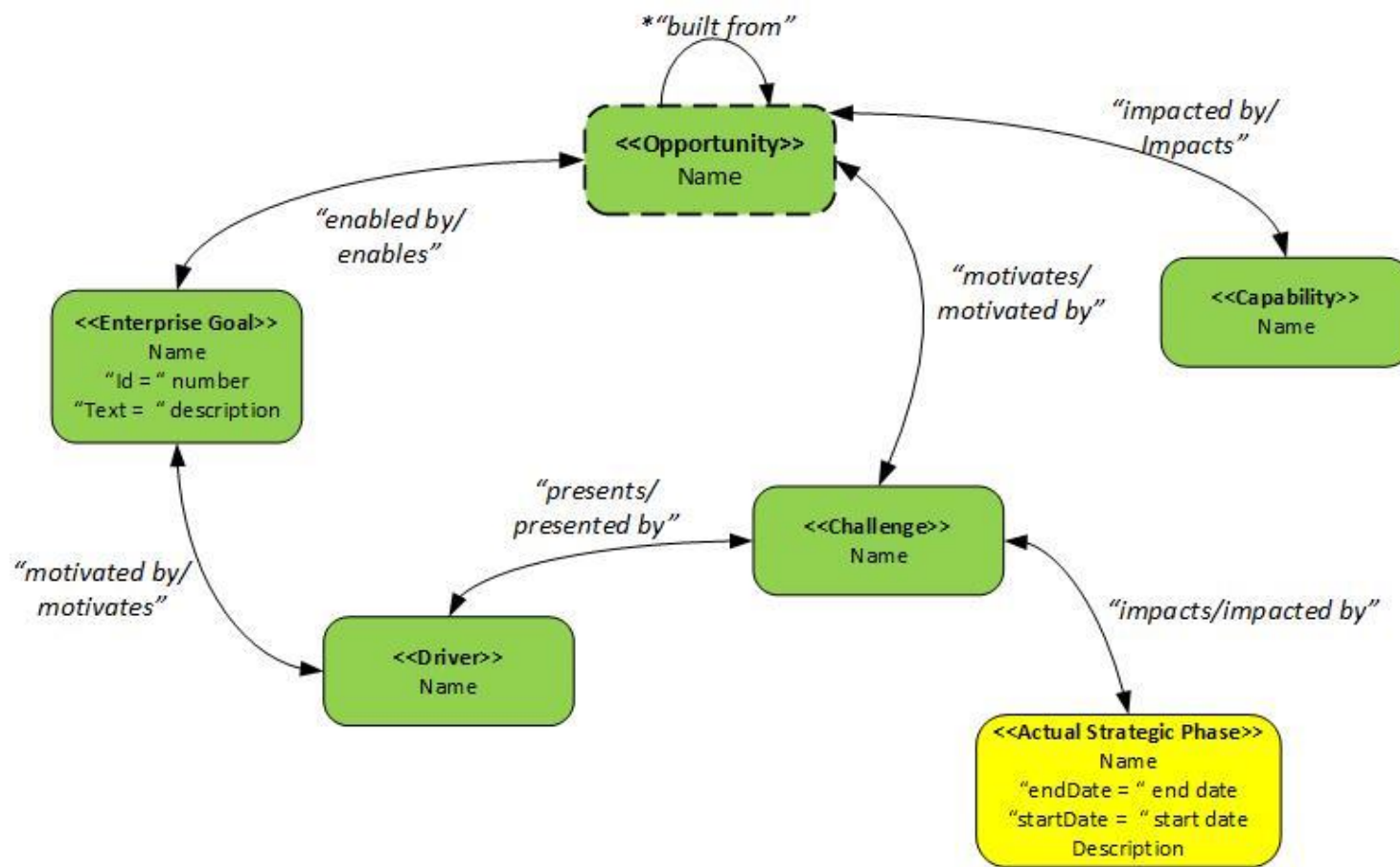


Architecture Traceability (Am-Tr)

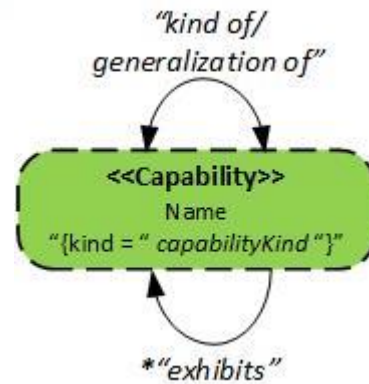


Strategic (St)

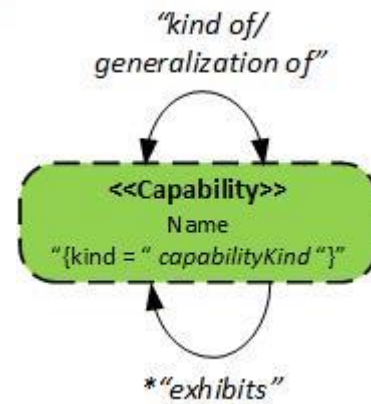
Strategic Motivation (St-Mv)



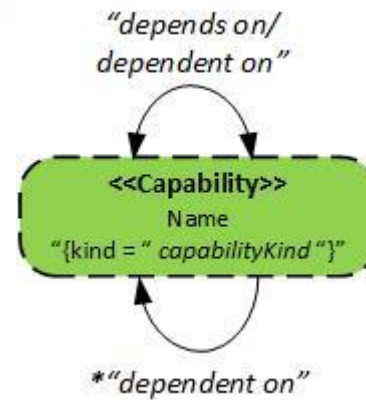
Strategic Taxonomy (St-Tx)



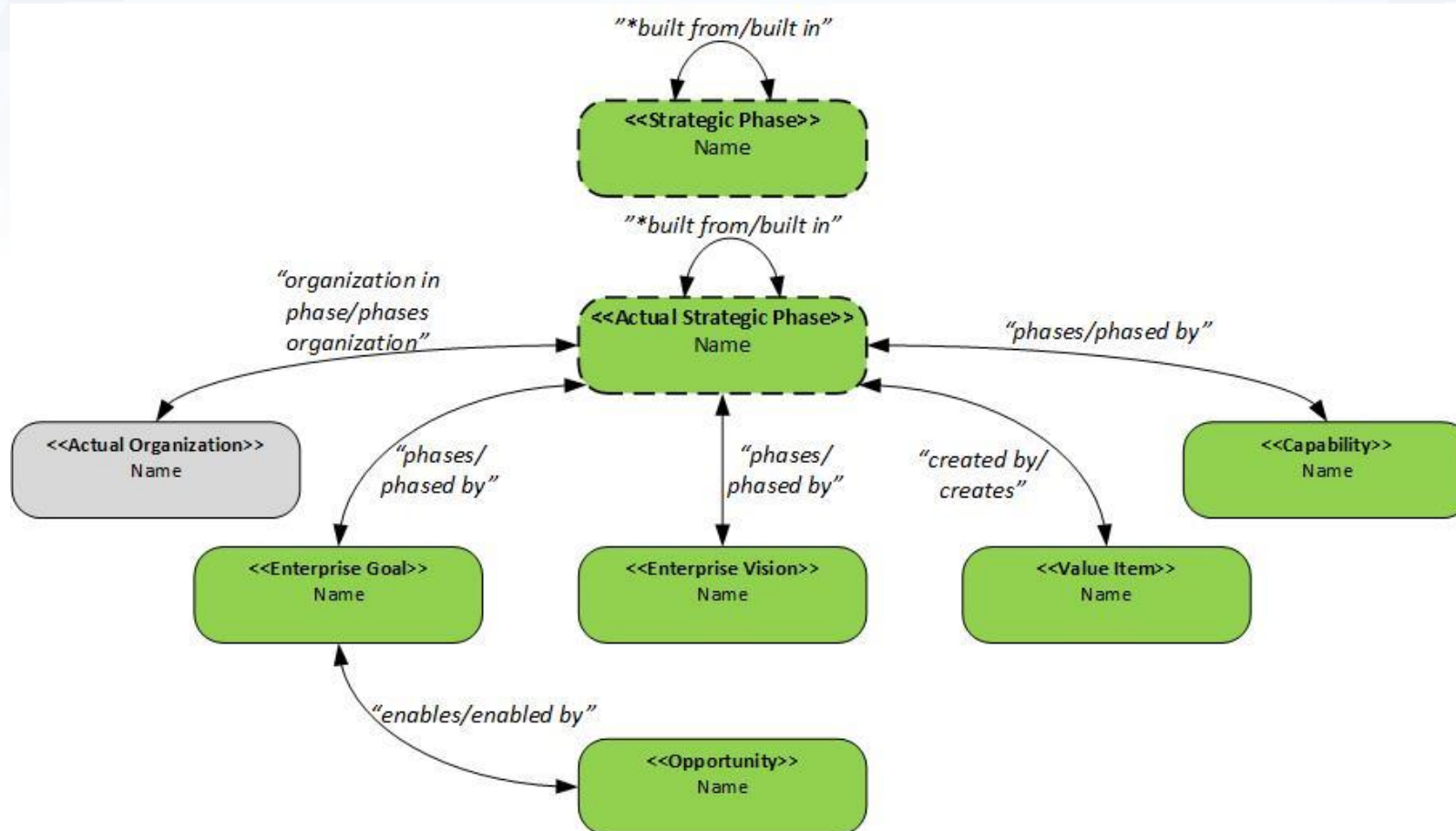
Strategic Structure (St-Sr)



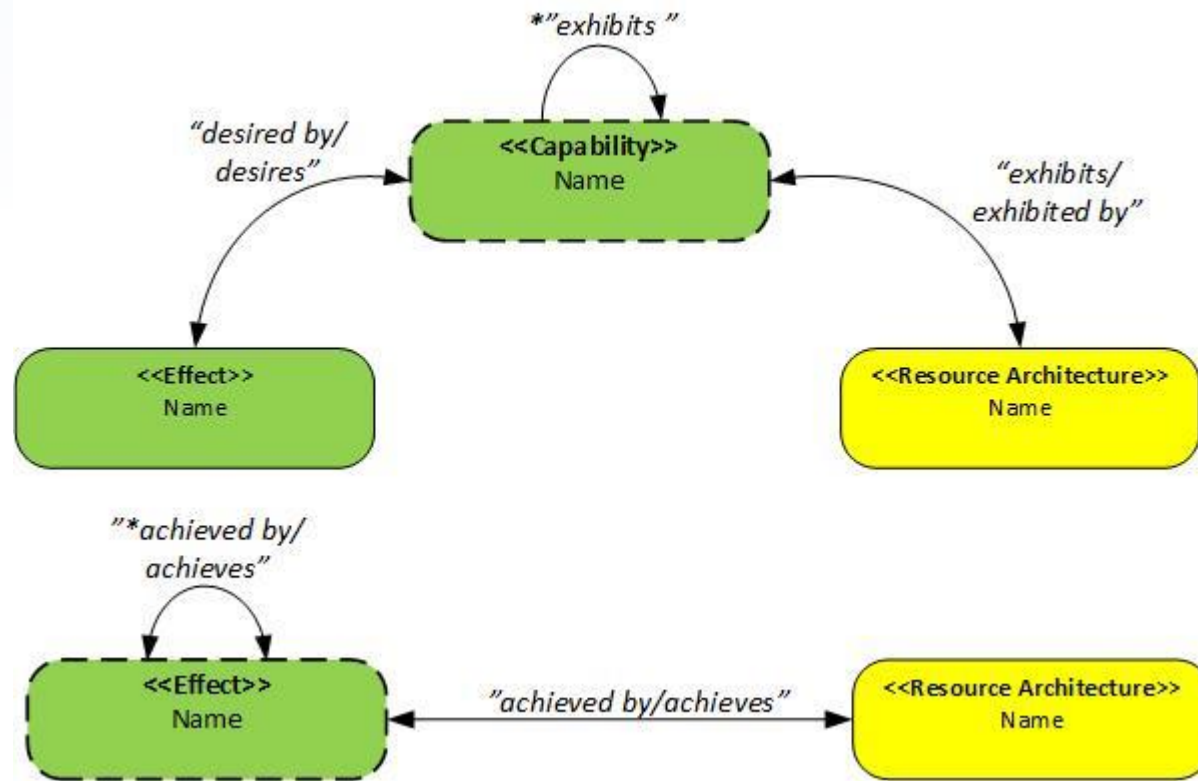
Strategic Connectivity (St-Cn)



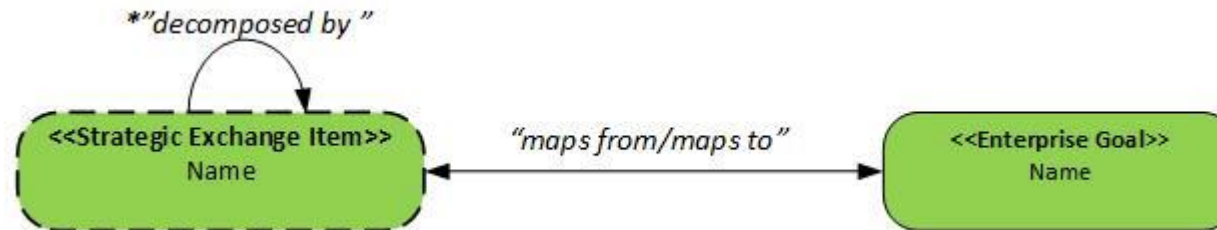
Strategic Processes (St-Pr)



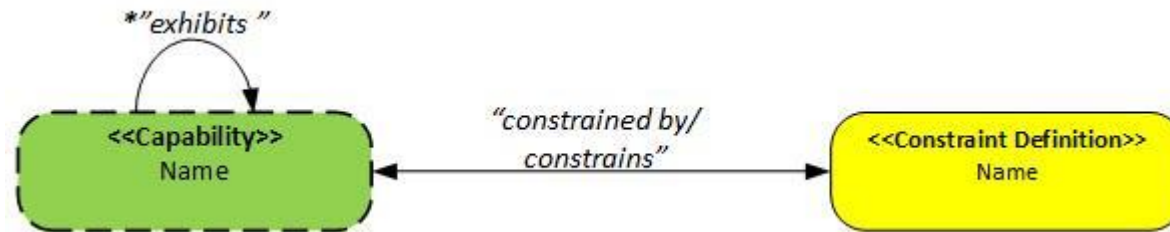
Strategic States (St-St)



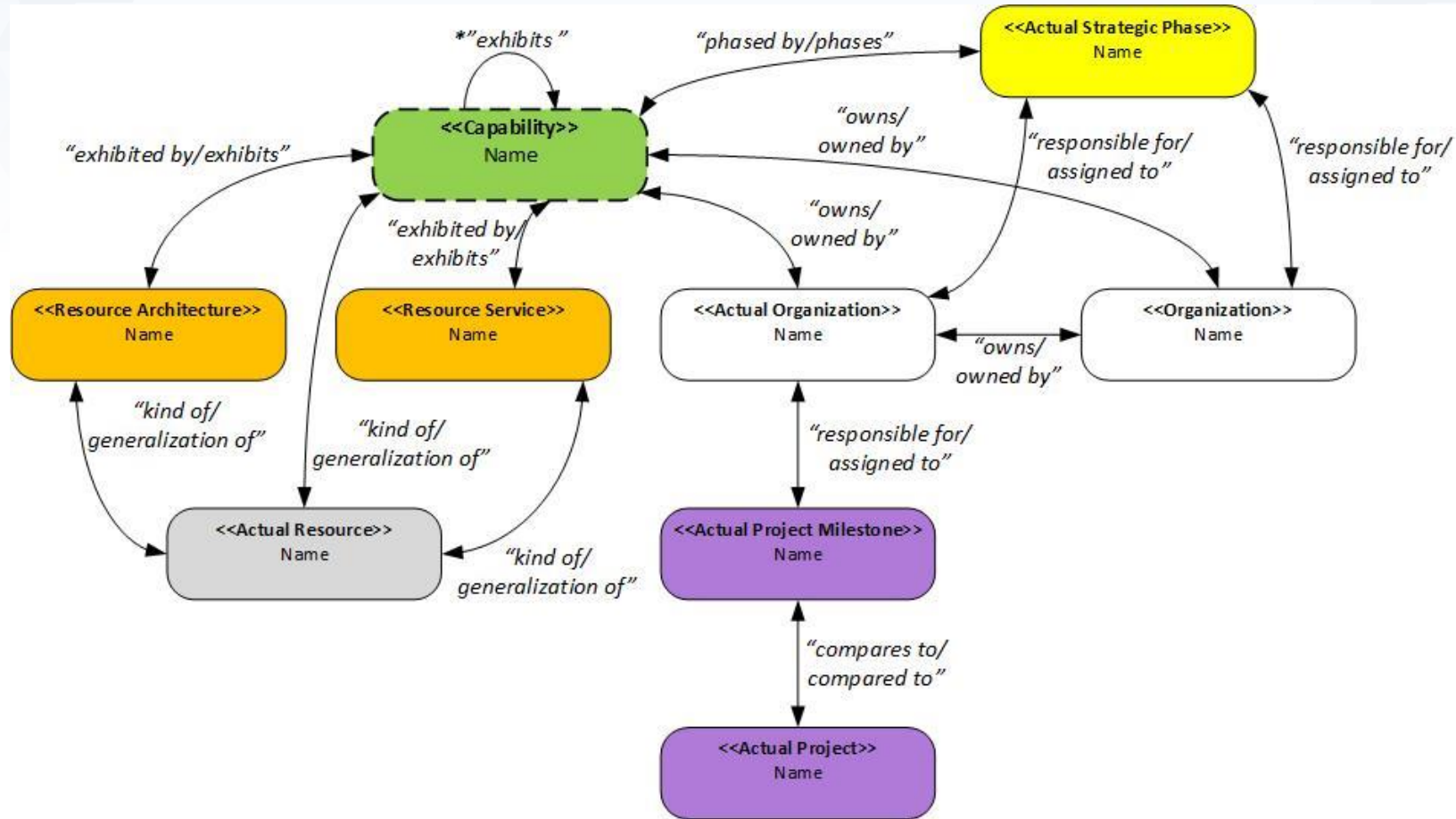
Strategic Information (St-If)



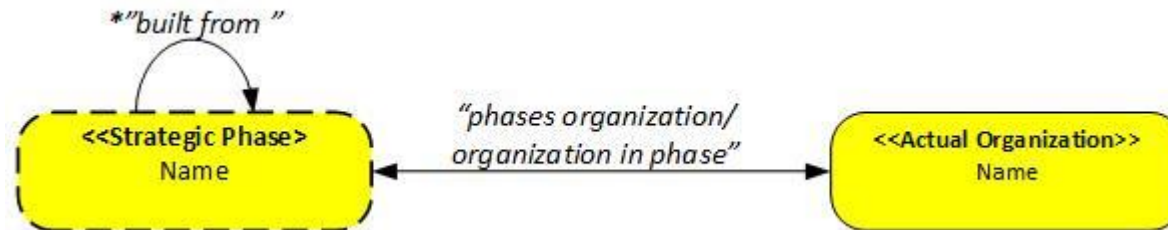
Strategic Constraints (St-Ct)



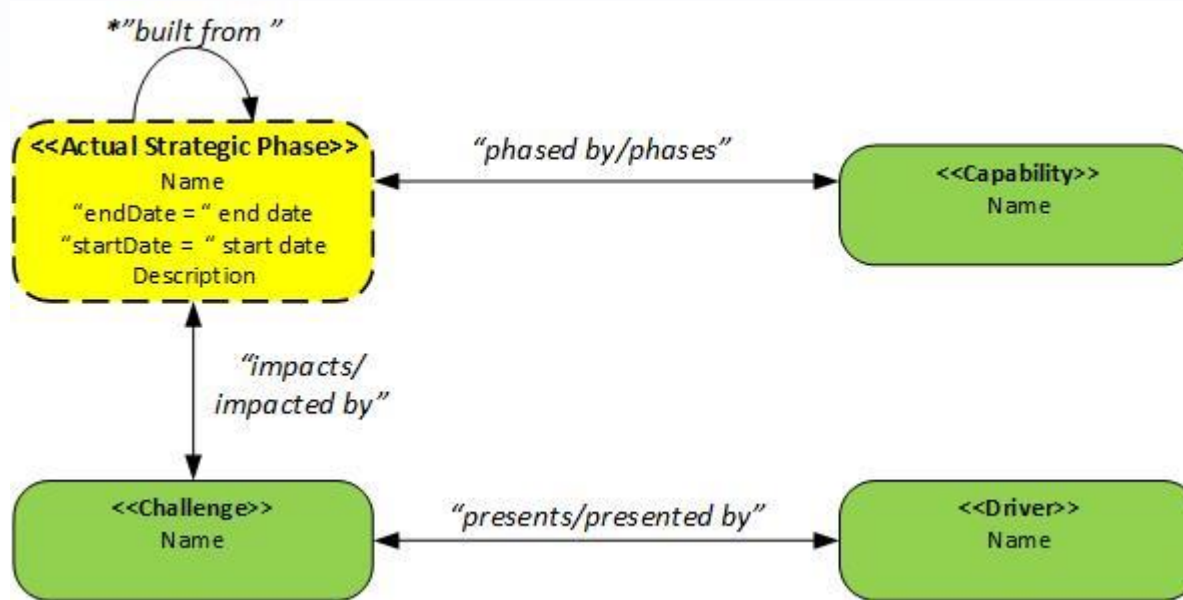
Strategic Roadmap (St-Rm-P)



Strategic Deployment (St-Rm-D)



Strategic Traceability (St-Tr)



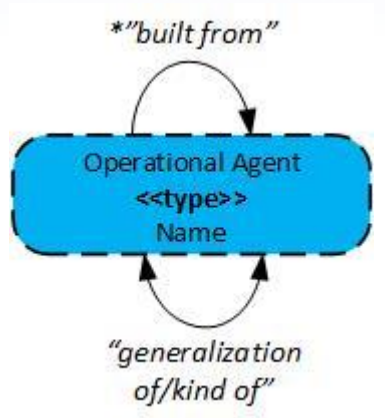
Operational (Op)

Operational Taxonomy (Op-Tx)

```
#define ASM_VMX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"  
static __always_inline unsigned long vmx_vmread(unsigned int rax, unsigned int rdx)  
{  
    unsigned long value;   
    __asm__ volatile ("vmx_vmread %0, %1, %2" : "=r"(value) : "r", "r"(rax, rdx));  
    return value;  
}  
lcs_solution lcs_half(const unsigned char *a, const string &b)  
{  
    lcs_solution result {}  
    const size_t size = a.size() * b.size();  
    for (size_t i = 0; i < size; ++i)  
        (result[i] = 0);  
    if (a.empty() || b.empty())  
        return result;  
    size_t a_size = a.size(), b_size = b.size();  
    for (size_t i = 0; i < a_size; ++i)  
        for (size_t j = 0; j < b_size; ++j)  
            result[i * b_size + j] = a[i] == b[j] ? result[i * b_size + j - 1] + 1 : 0;  
    return result;  
}
```

```
Request  
Success  
Merge  
Unmerge  
UseSuccess  
UseFailure  
ViewData  
Value  
Control  
Request  
Success  
Unmerge  
UseSuccess  
UseFailure  
ViewData  
Output  
Input  
b Point  
Request  
Success  
Unmerge  
UseSuccess  
UseFailure  
ViewData  
Request  
Success  
Unmerge  
UseSuccess  
UseFailure  
ViewData
```

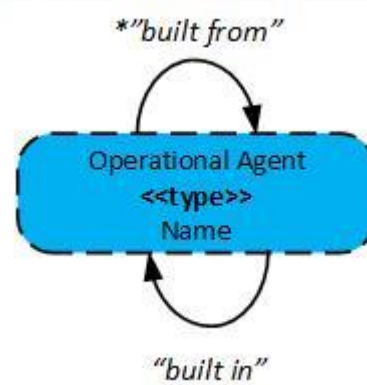
```
%include "win32n.inc"  
extern MessageBoxA  
import MessageBoxA user32.dll  
extern WINAPI DefProc  
extern WINAPI DefProc
```



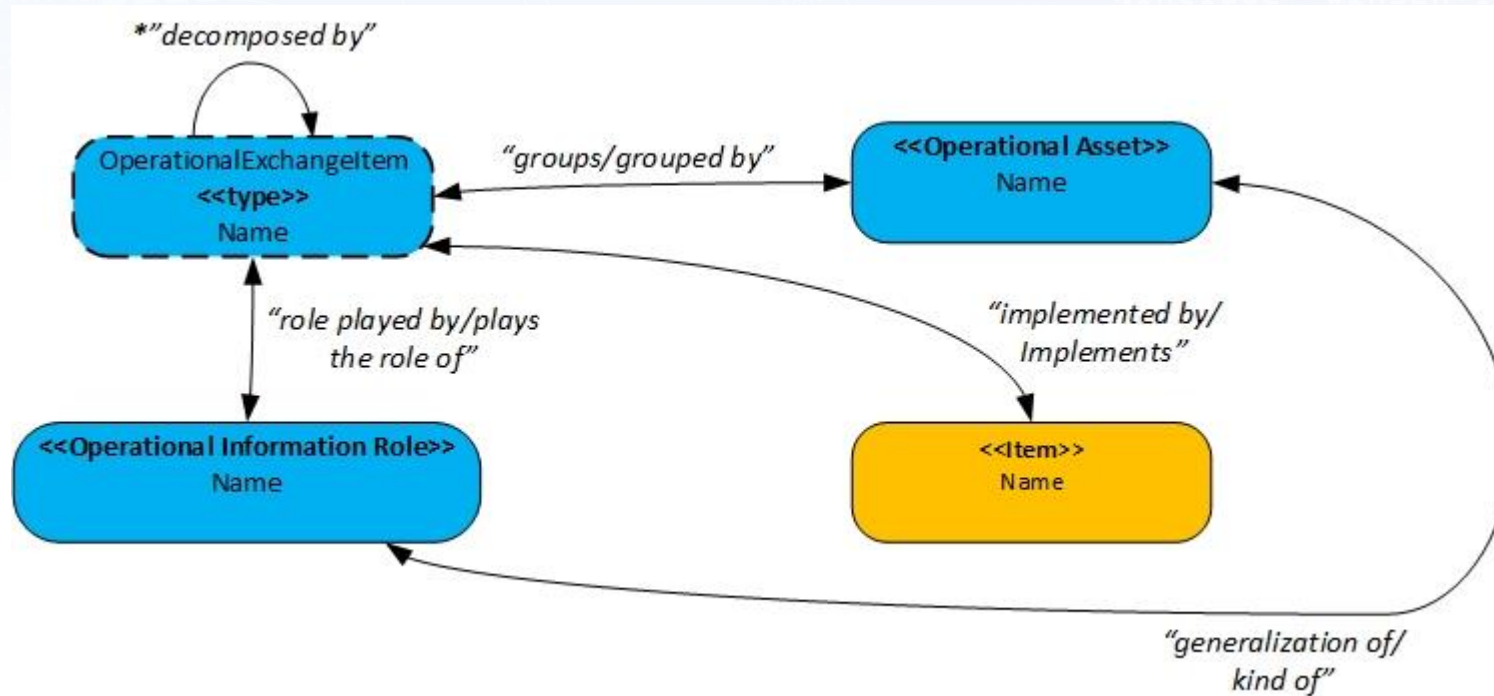
Operational Structure (Op-Sr)

```
#define ASM_VRX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"  
static __always_inline unsigned long vmread(unsigned long rdx, unsigned long rax)  
{  
    unsigned long value;   
    __asm__ volatile ("vmread %0,%1,%2" : "=r" (value) : "r" (rdx), "r" (rax));  
    return value;  
}  
lcs_solution lcs_half(const unsigned char *a, const string &b)  
{  
    lcs_solution result {}  
    const size_t asz = a.size(), bsz = b.size();  
    for (size_t i = 0; i < asz; ++i)  
        for (size_t j = 0; j < bsz; ++j)  
            result[i][j] = 0;  
    result[0][0] = 1;  
    for (size_t i = 1; i < asz; ++i)  
        for (size_t j = 1; j < bsz; ++j)  
            result[i][j] = (a[i-1] == b[j-1]) ? result[i-1][j-1] + 1 : max(result[i-1][j], result[i][j-1]);  
    return result[asz-1][bsz-1];  
}
```

```
%include "win32n.inc"  
extern MessageBoxA  
import MessageBoxA user32.dll  
extern WINAPI DefProc  
extern WINAPI DefProcA  
extern WINAPI DefProcW  
extern WINAPI DefProcWEx
```



Operational Information (Op-If)



Operational Traceability (Op-Tr)

```
#define ASM_VMX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"
```

```
static __always_inline unsigned long vmx_vmread(unsigned int vmcs, unsigned int field)
```

```
{  
    unsigned long value;
```

```
    __asm__ volatile ("vmx_vmread %0, %1, %2" : "=r" (value) : "r" (vmcs), "r" (field));
```

```
    return value;
```

```
}  
lcs_solution lcs_half(const unsigned char * const string, int)
```

```
lcs_solution result (0);
```

```
const size_t size = a.size() + b.size();
```

```
for (size_t i = 0; i < size; ++i)
```

```
{  
    for (size_t j = 0; j < b.size(); ++j)
```

```
{  
        if (a[i-j] + b[j] > result.value)
```

```
            result.value = a[i-j] + b[j];
```

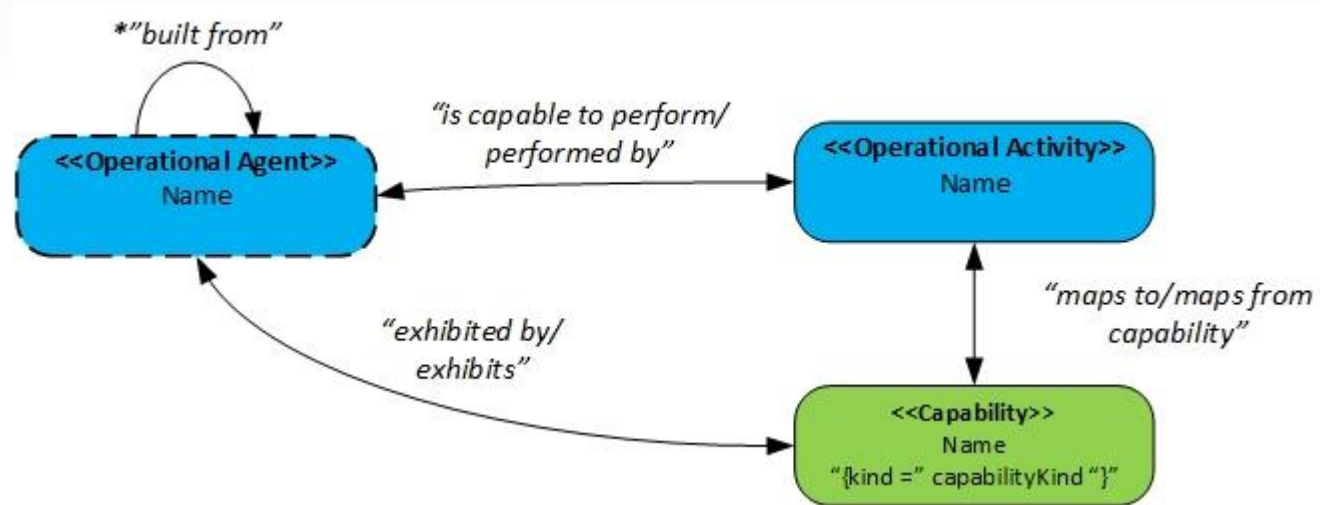
```
    }  
}
```

```
%include "win32n.inc"
```

```
extern MessageBoxA
```

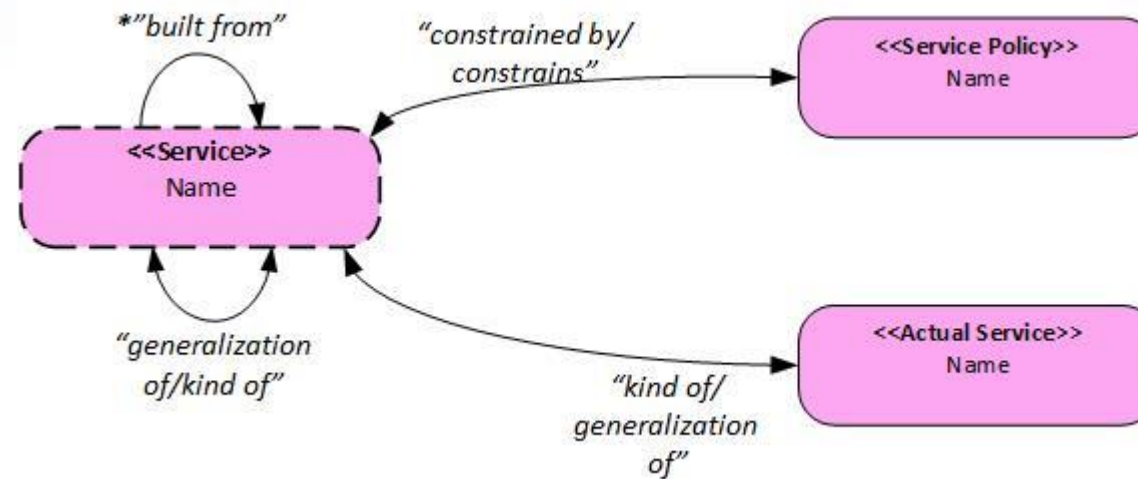
```
!import MessageBoxA user32.dll
```

```
!include "user32.inc"
```

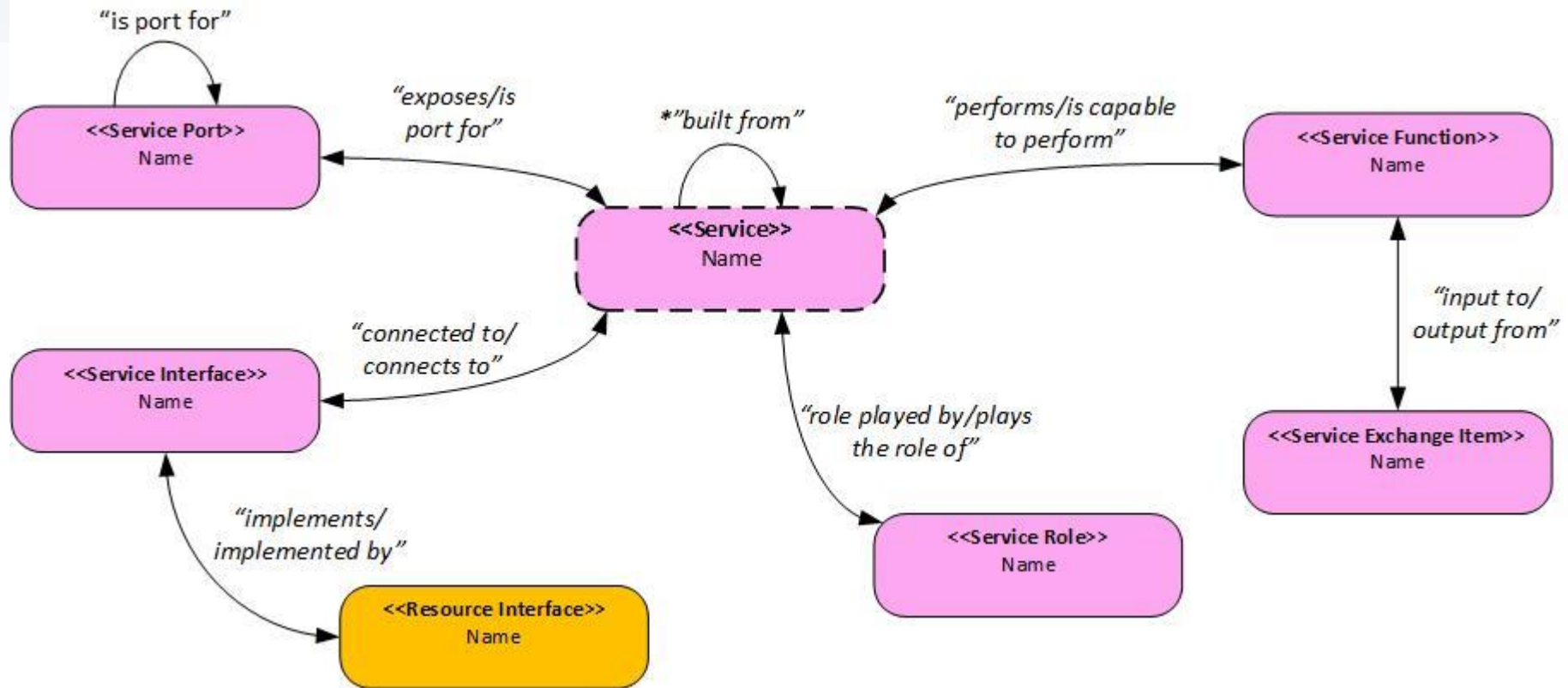


Services (Sv)

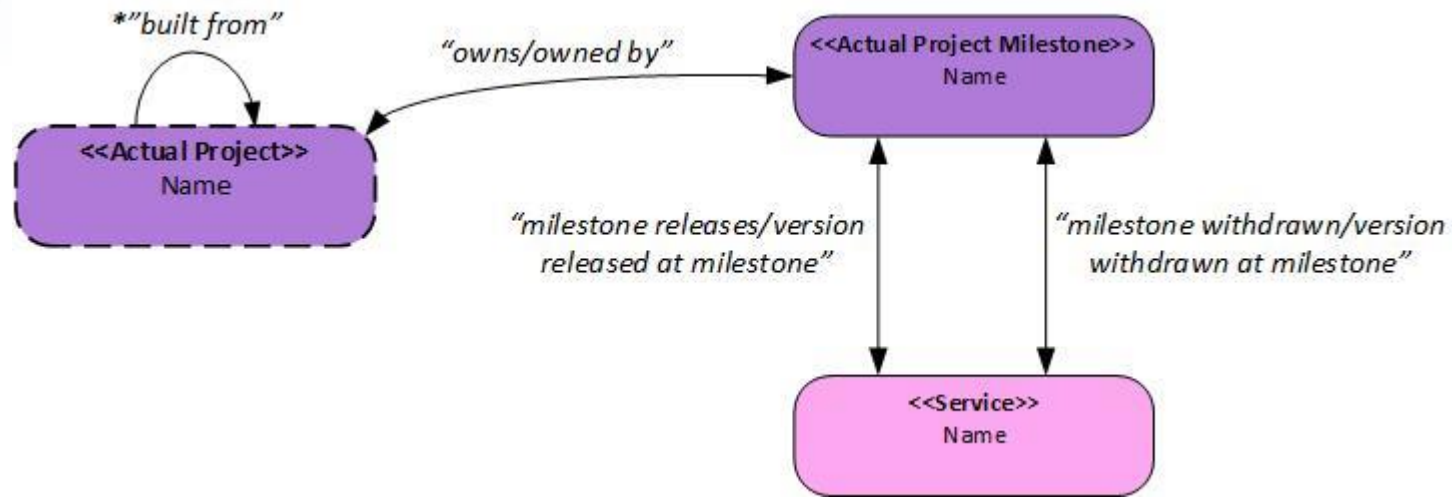
Services Taxonomy (Sv-Tx)



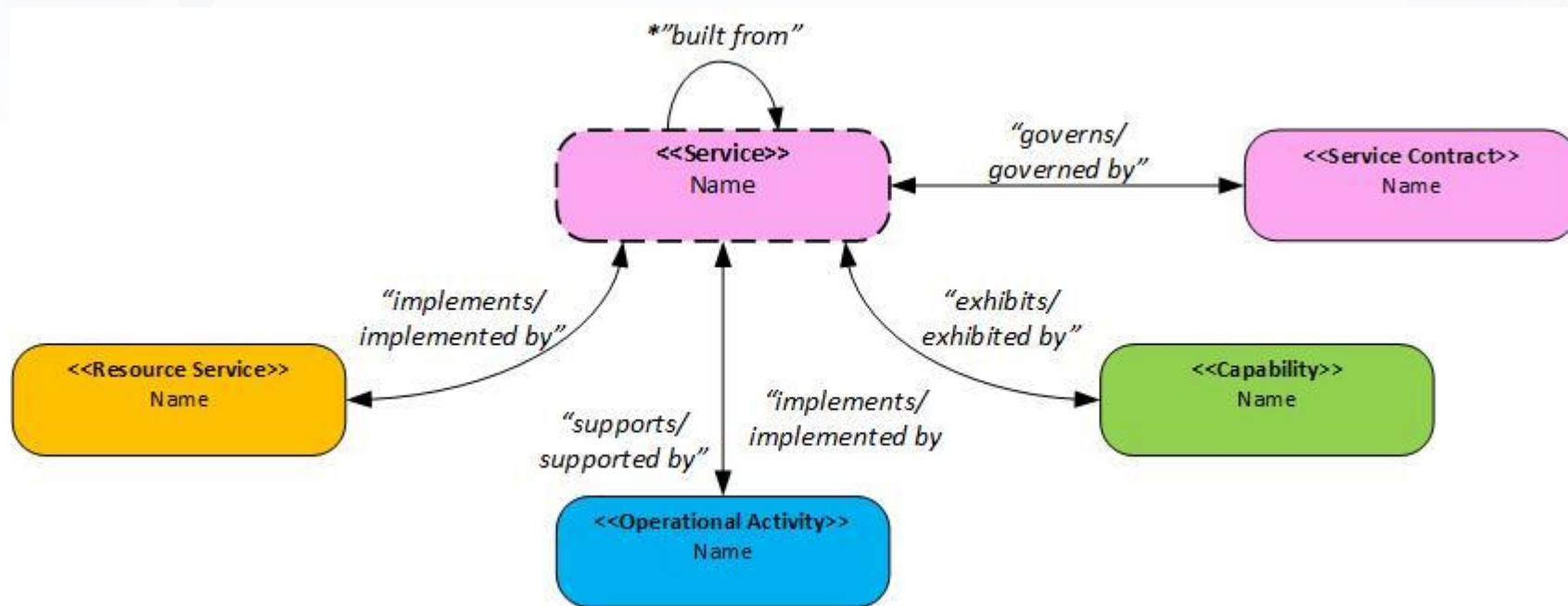
Services Structure (Sv-Sr)



Services Roadmap (Sv-Rm)



Services Traceability (Sv-Tr)

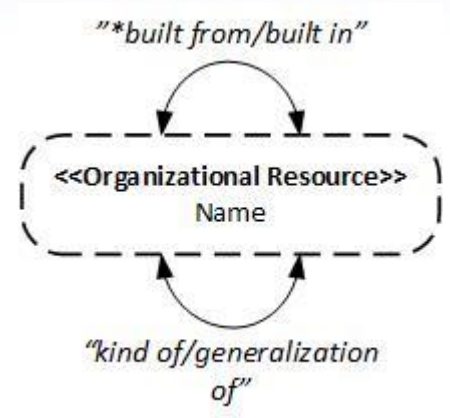


Personnel (Ps)

Personnel Taxonomy (Ps-Tx)

```
#define ASM_VMX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"  
static __always_inline unsigned long vmx_vmread(unsigned int reg, unsigned int *value)  
{  
    lcs_solution lcs_half(const char *str, const string &b)  
    lcs_solution result (0);  
    const size_t size = a.size() - b.size();  
    for (size_t i = 0; i < size; ++i)  
        for (size_t j = 0; j < b.size(); ++j)  
            if (a[i+j] != b[j])  
                return 0;  
    return a[i];  
}
```

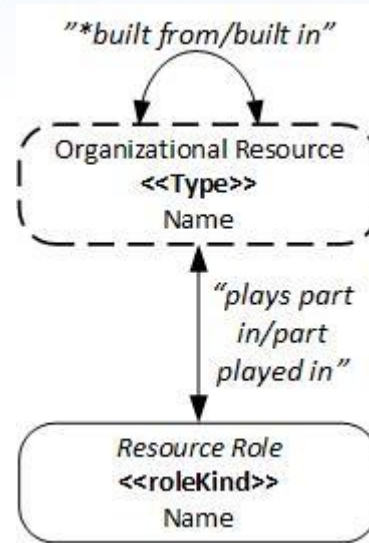
```
%include "win32n.inc"  
extern MessageBoxA  
import MessageBoxA user32.dll  
extern WINAPI DefProc  
extern WINAPI DefProc
```



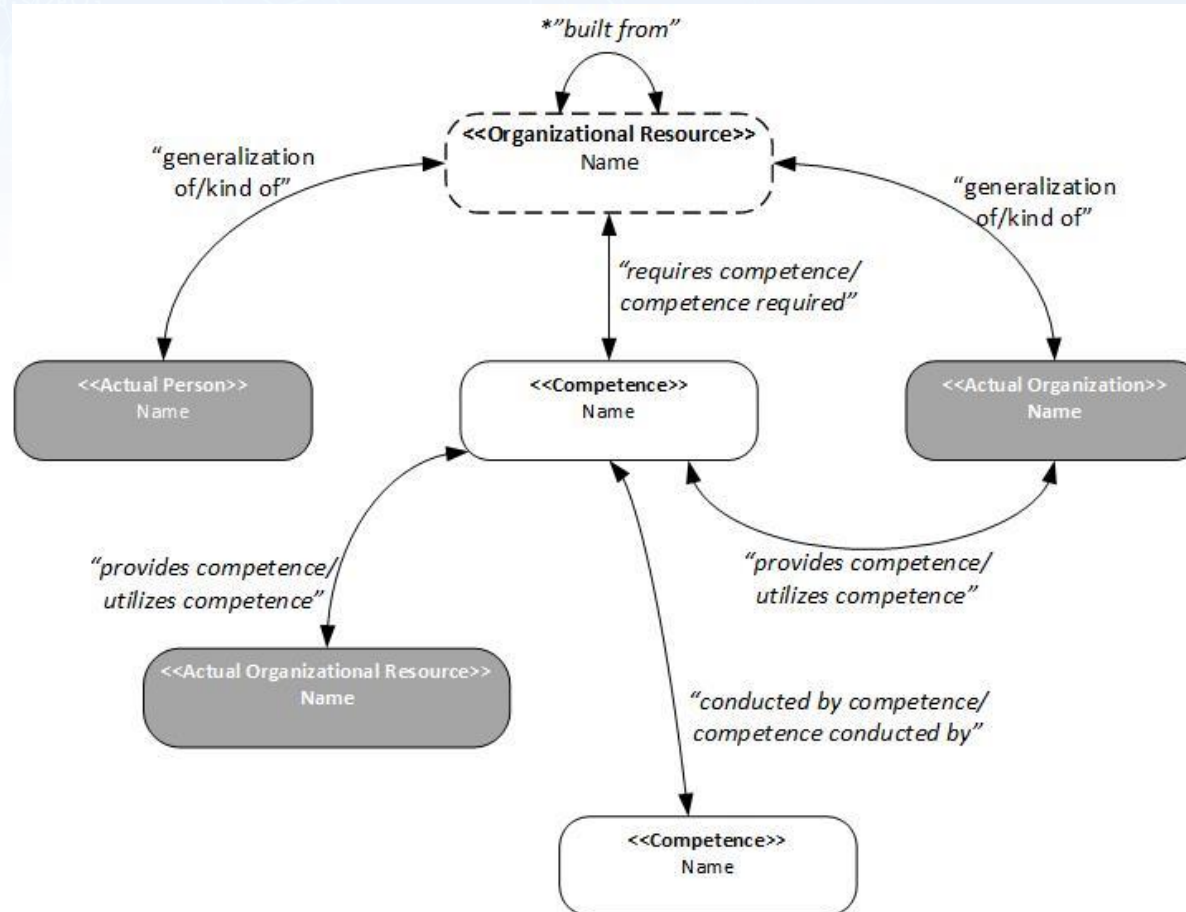
Personnel Structure (Ps-Sr)

```
#define ASM_VMX_VMREAD_RDX_RAX    ".byte 0xD1, 0x7A, 0xD0"  
static __always_inline unsigned long vmx_vmread_rdx_rax(struct vmx_vmcs *vmcs, unsigned int rdx_rax) {  
    unsigned long value;   
    vmcs->cs_solution = cs_half(const char *cs_half, const string kb)  
    cs_solution result ();  
    const size_t size = a.size() - b.size();  
    for (size_t i = 0; i < size; ++i) {  
        (on (size_t) i = 0; i < size; ++i)  
        if (a[i] != b[i]) {  
            return 0;  
        }  
    }  
    return 1;  
}
```

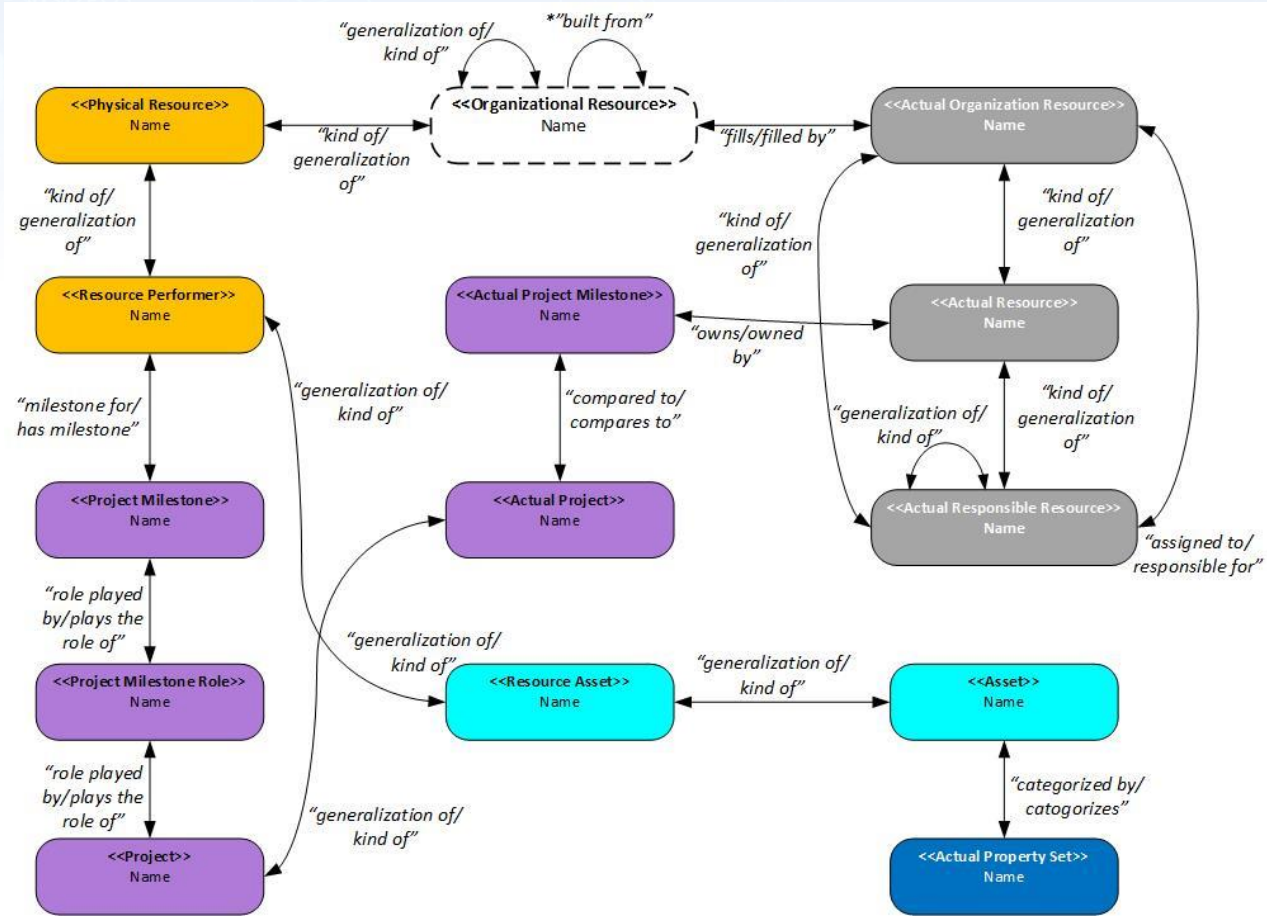
```
%include "win32n.inc"  
extern MessageBoxA  
import MessageBoxA user32.dll  
extern WINAPI_FAMILY_NAME  
#define WINAPI_FAMILY WINAPI_FAMILY_DESKTOP_APP
```



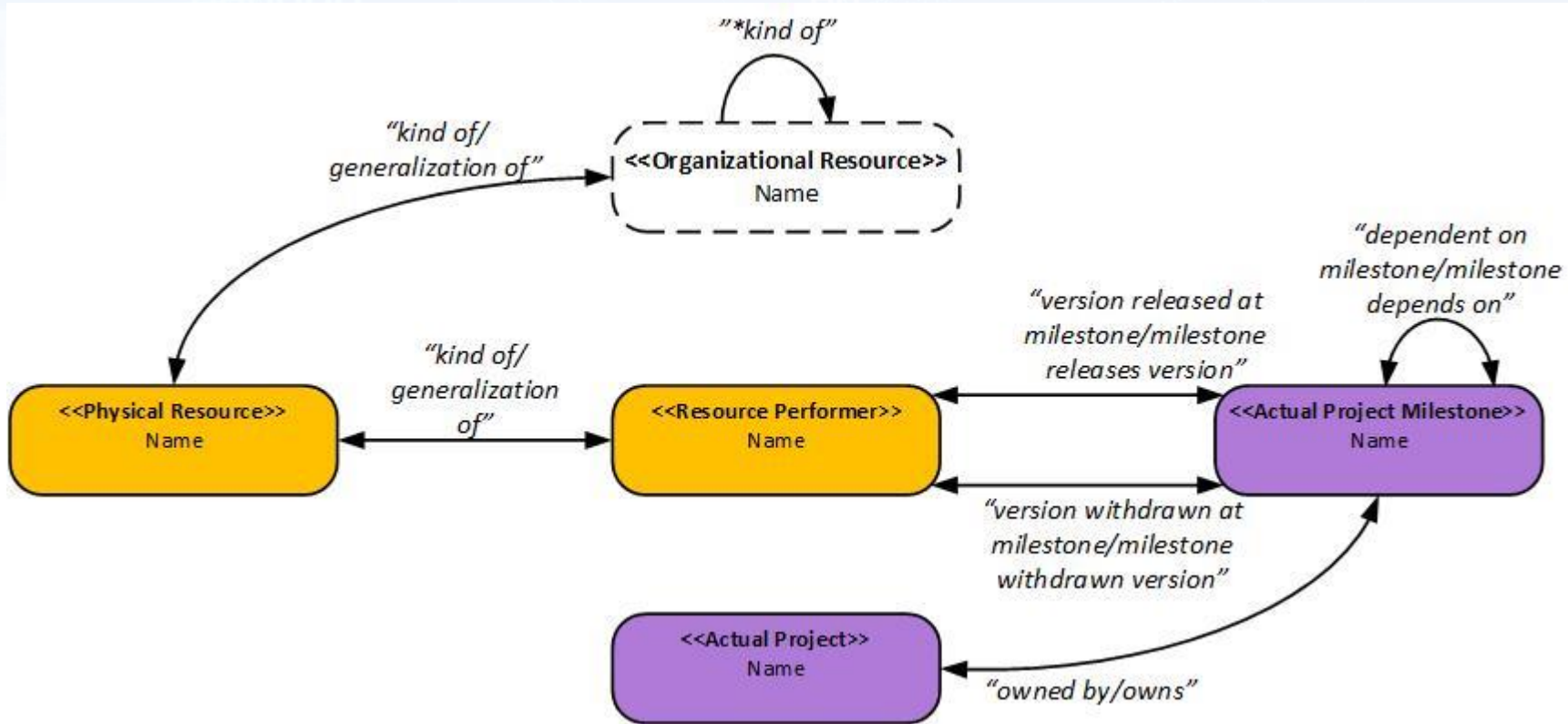
Competence, Drivers, Performance (Ps-Ct)



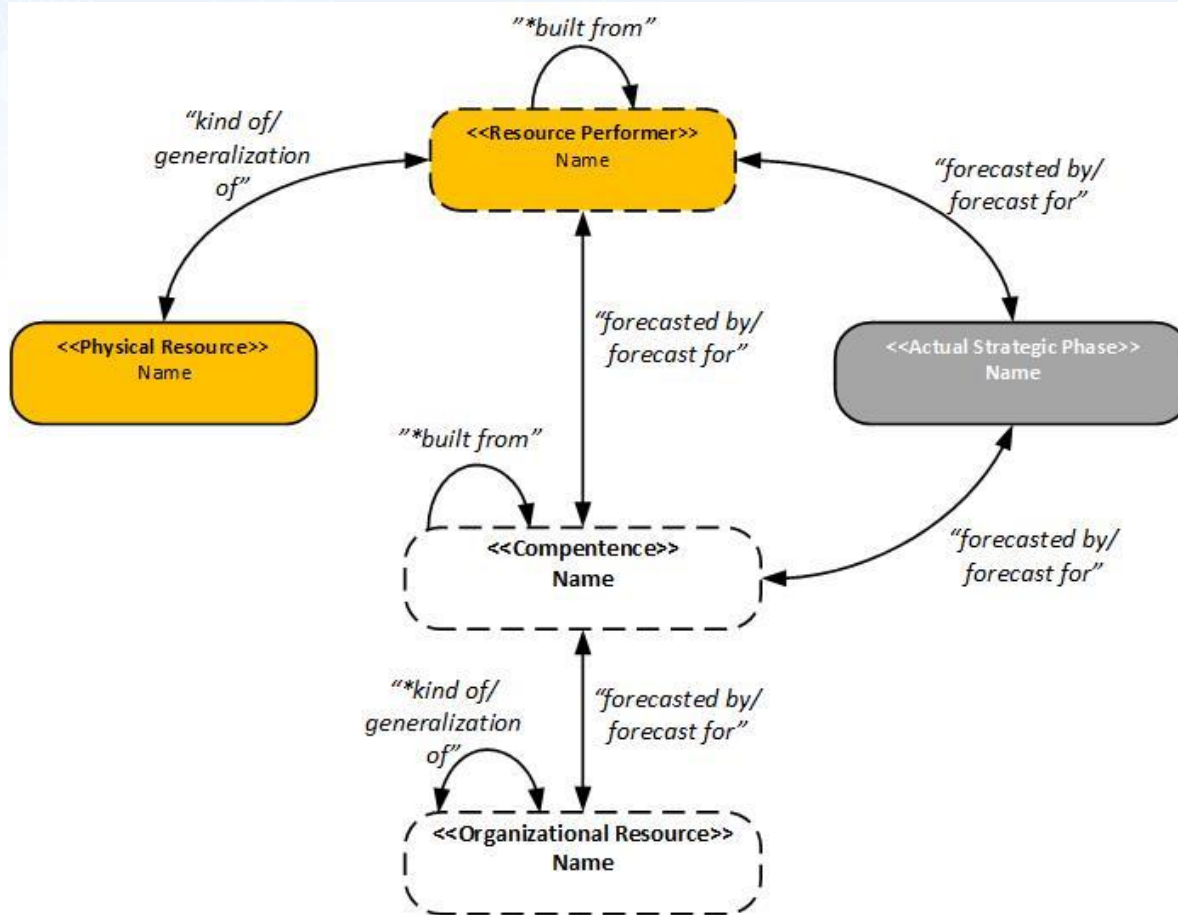
Personnel Availability (Ps-Rm-A)



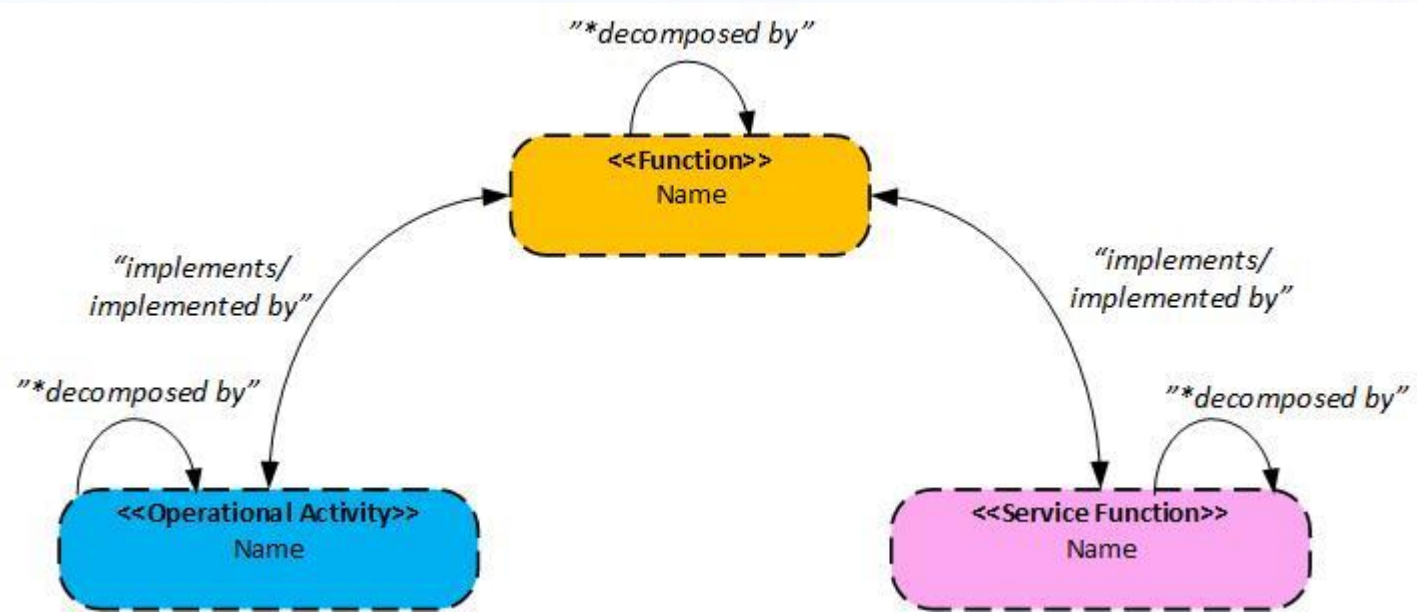
Personnel Evolution (Ps-Rm-E)



Personnel Forecast (Ps-Rm-F)

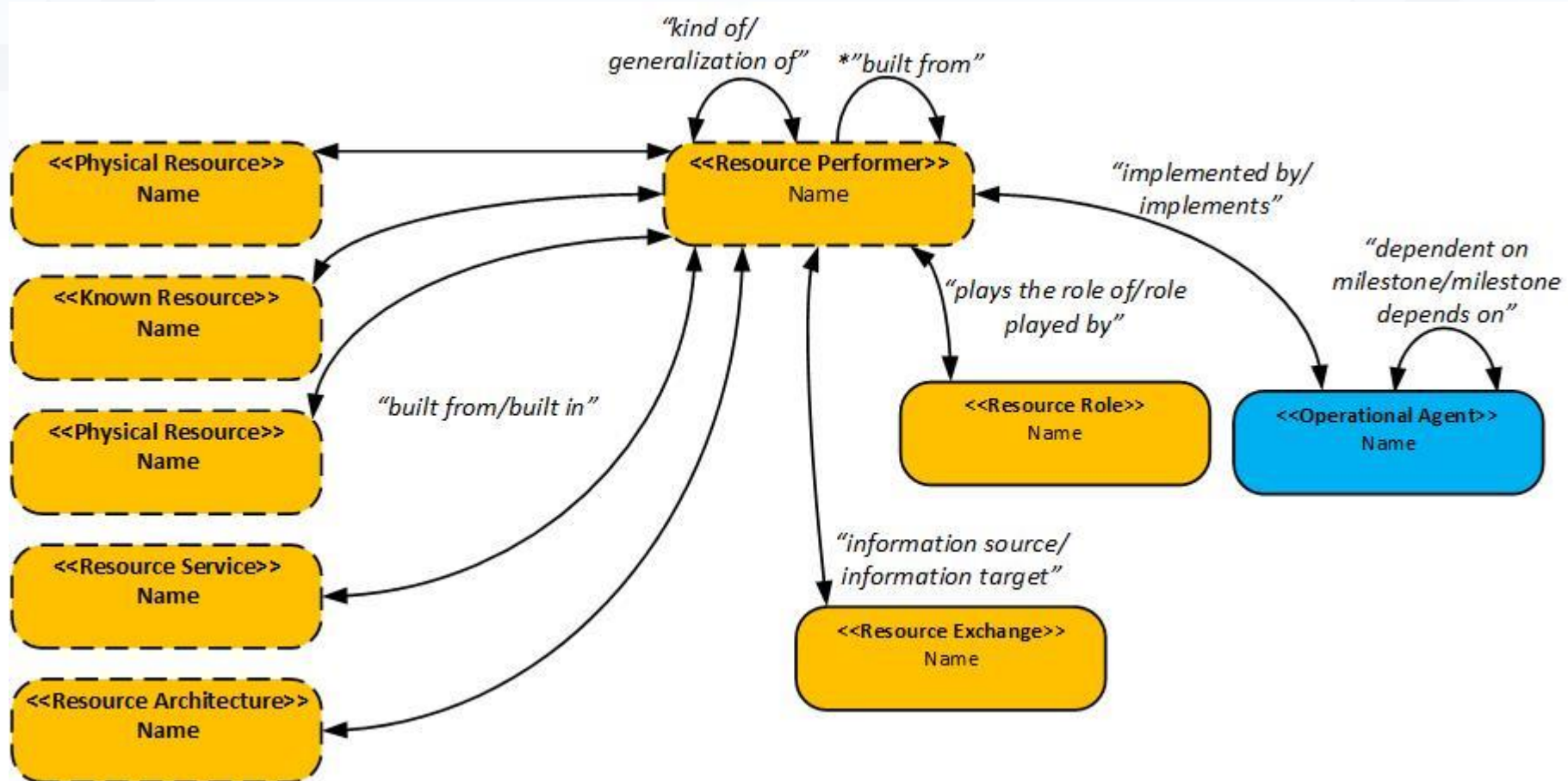


Personnel Traceability (Ps-Tr)

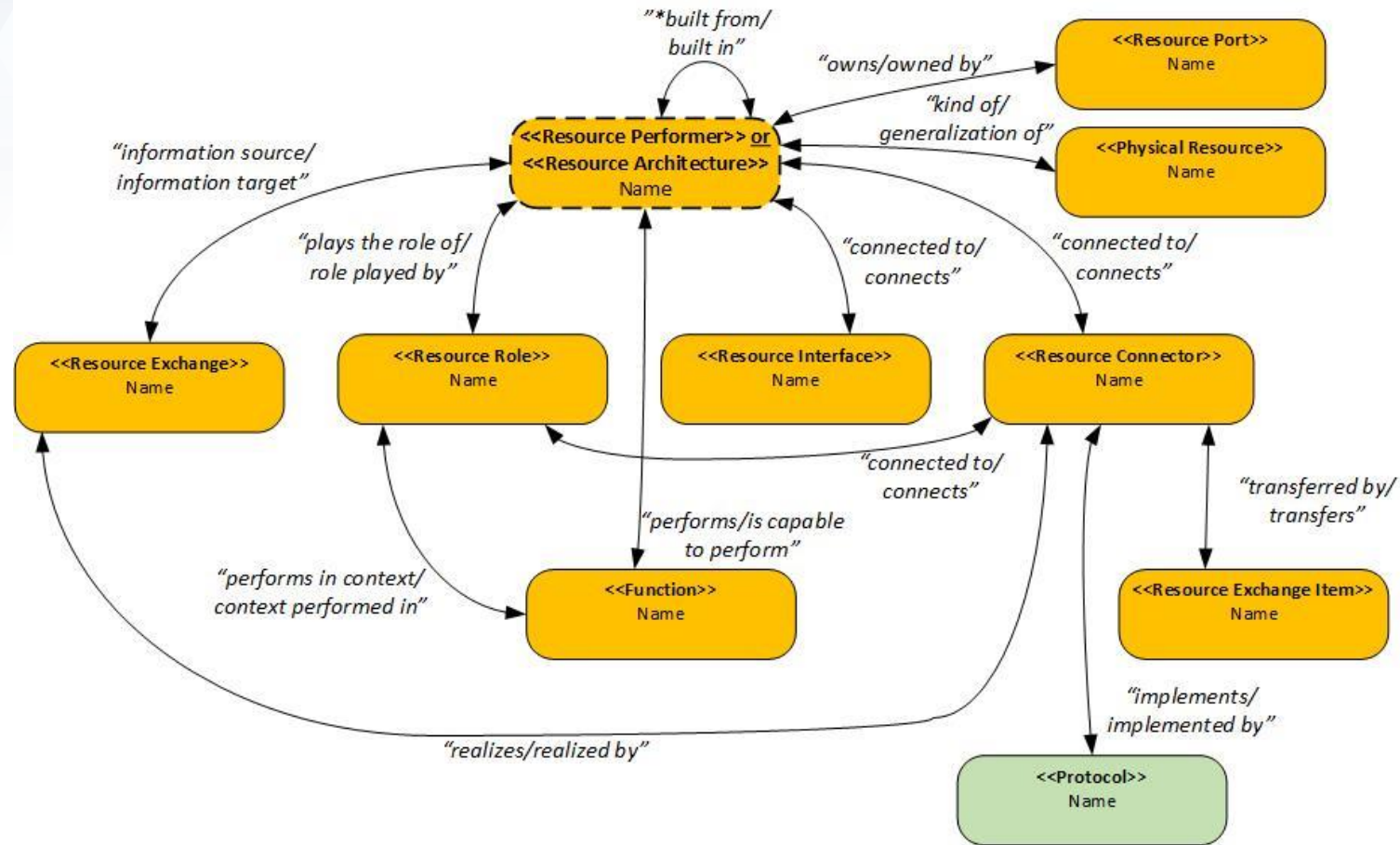


Resources (Rs)

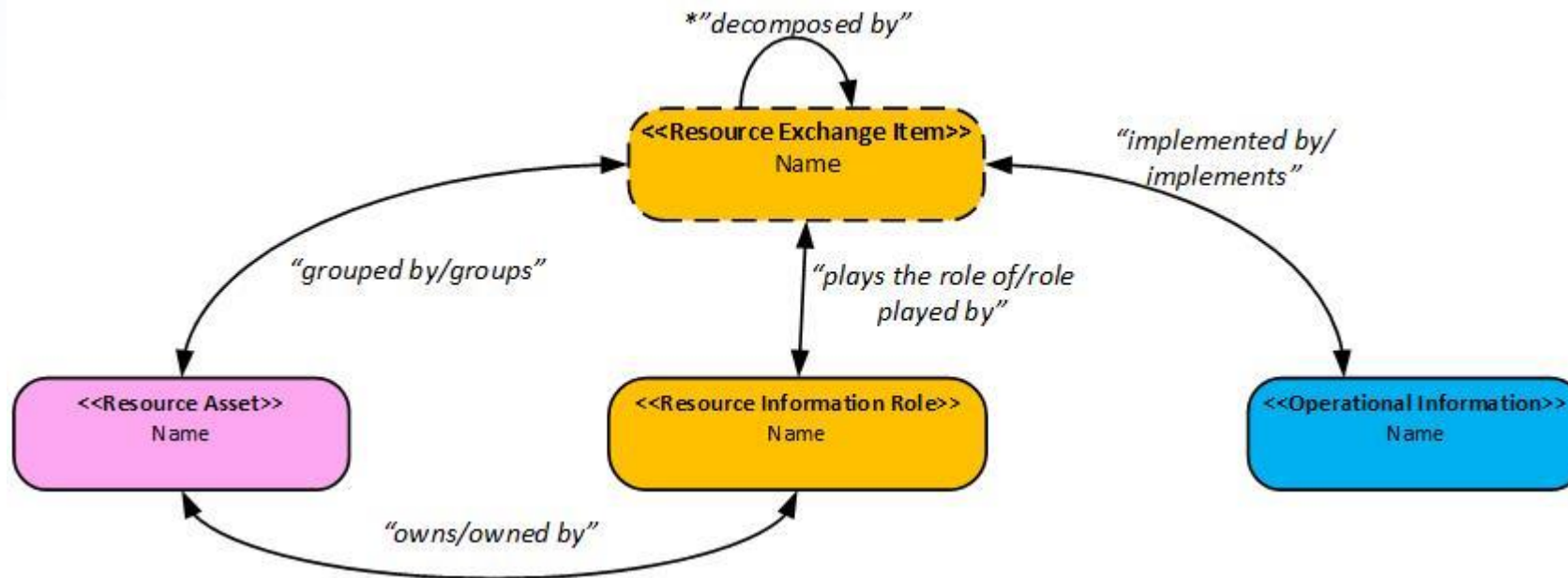
Resources Taxonomy (Rs-Tx)



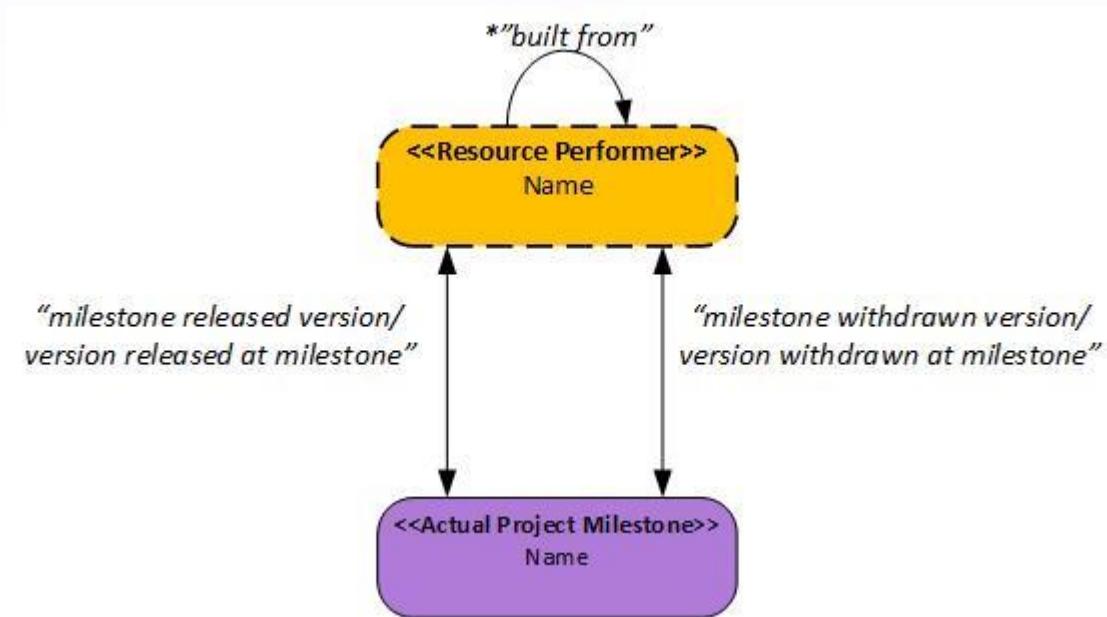
Resources Structure (Rs-Sr)



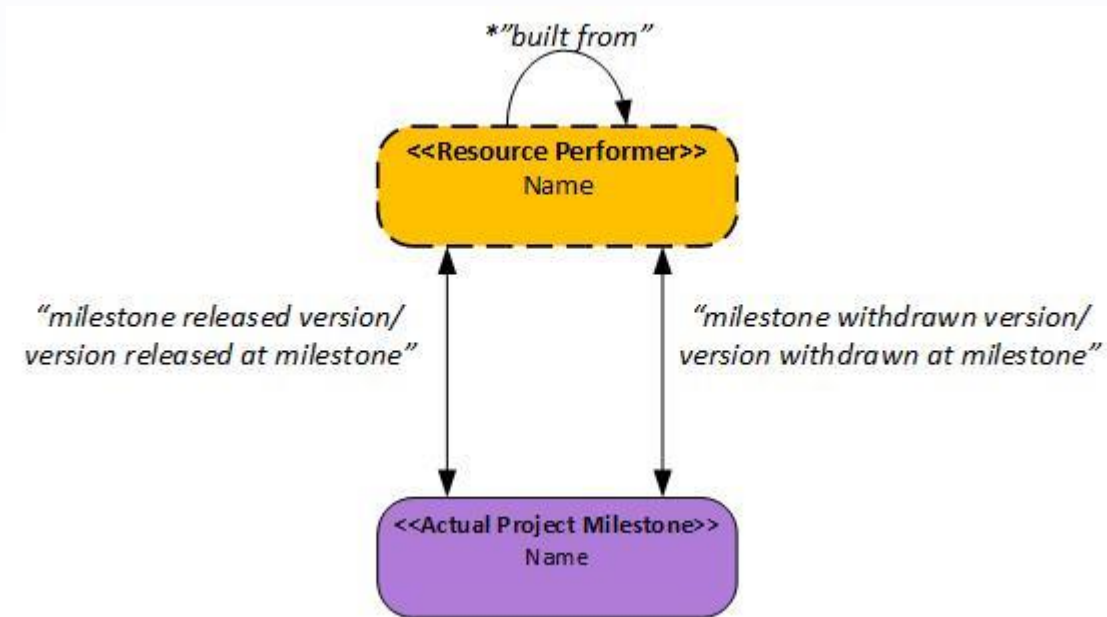
Resources Information Model (Rs-If)



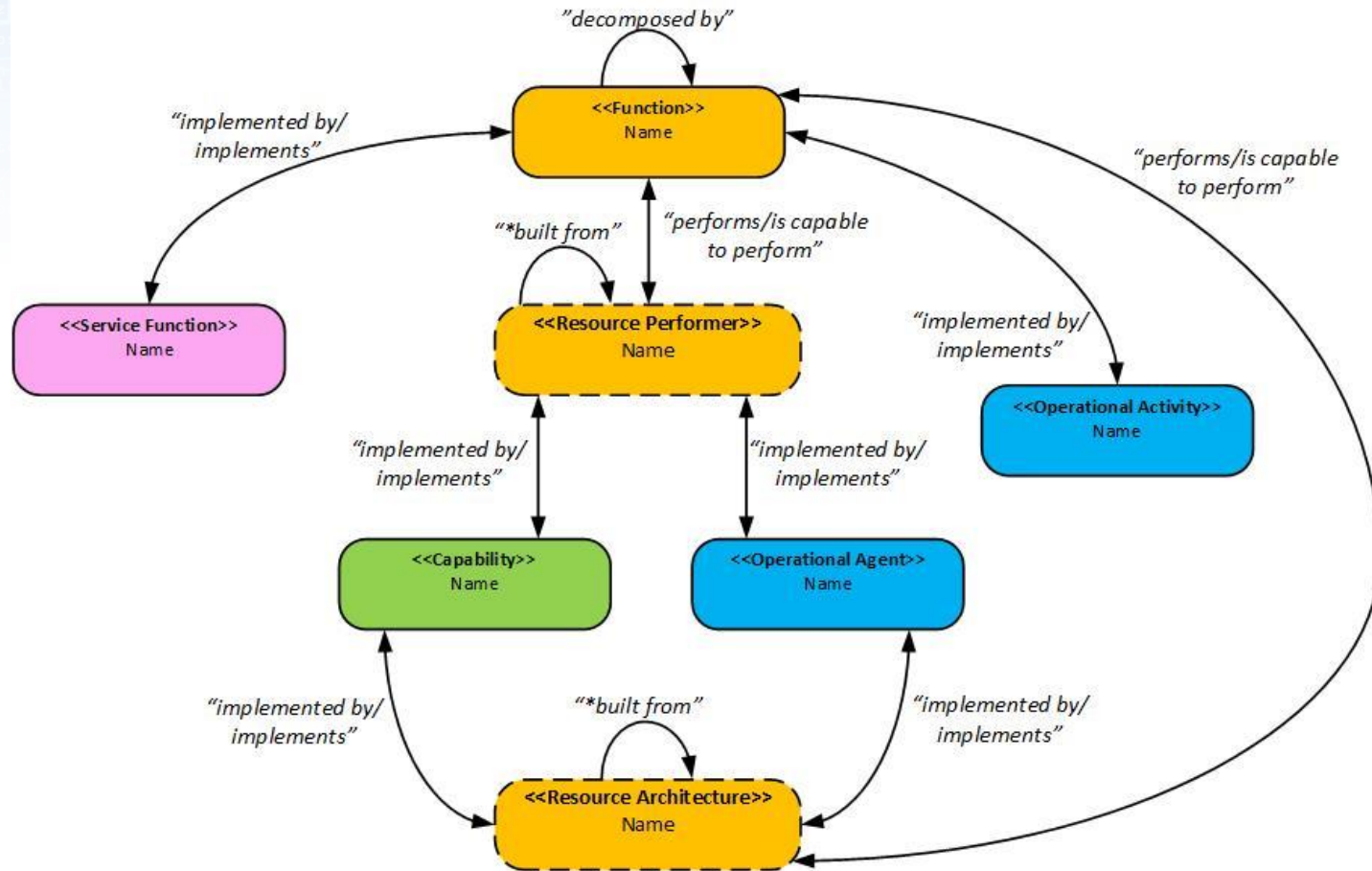
Resources Evolution (Rs-Rm-E)



Resources Forecast (Rs-Rm-F)

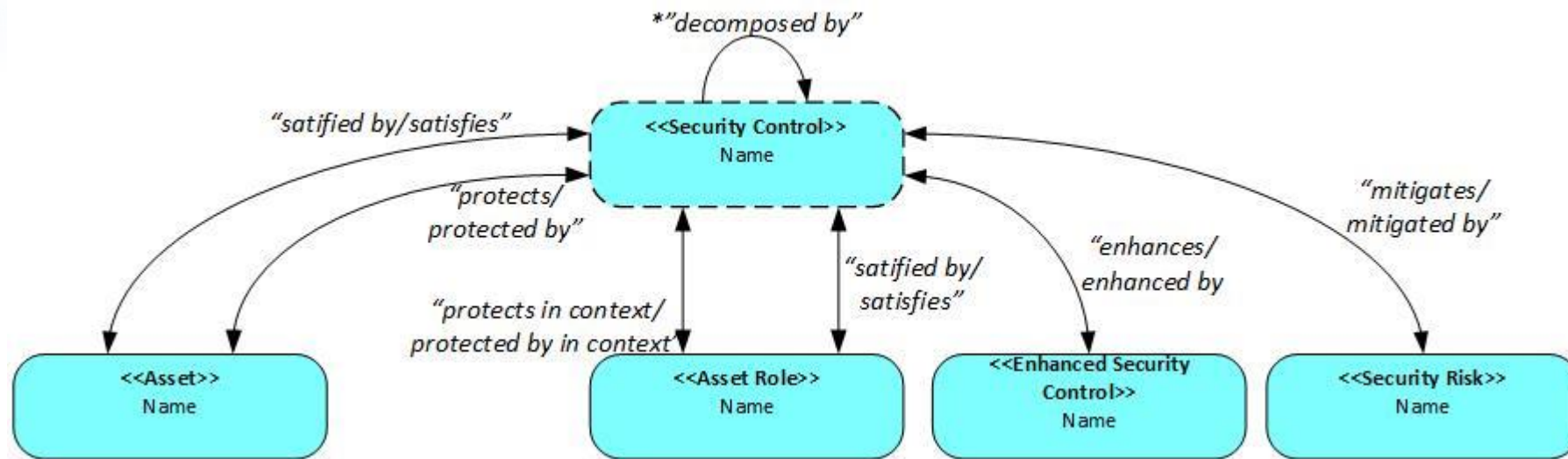


Resources Traceability (Rs-Tr)

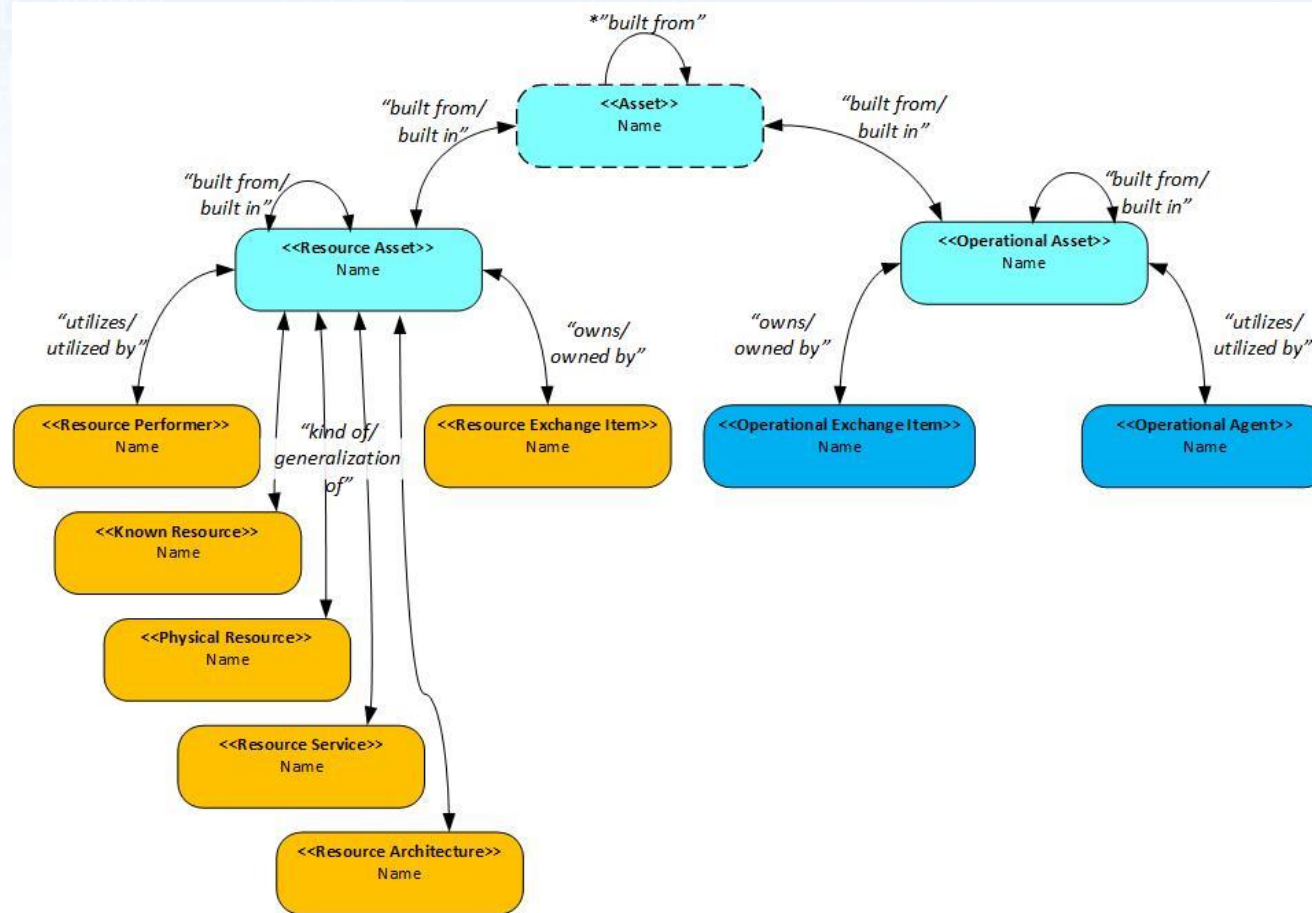


Security (Sc)

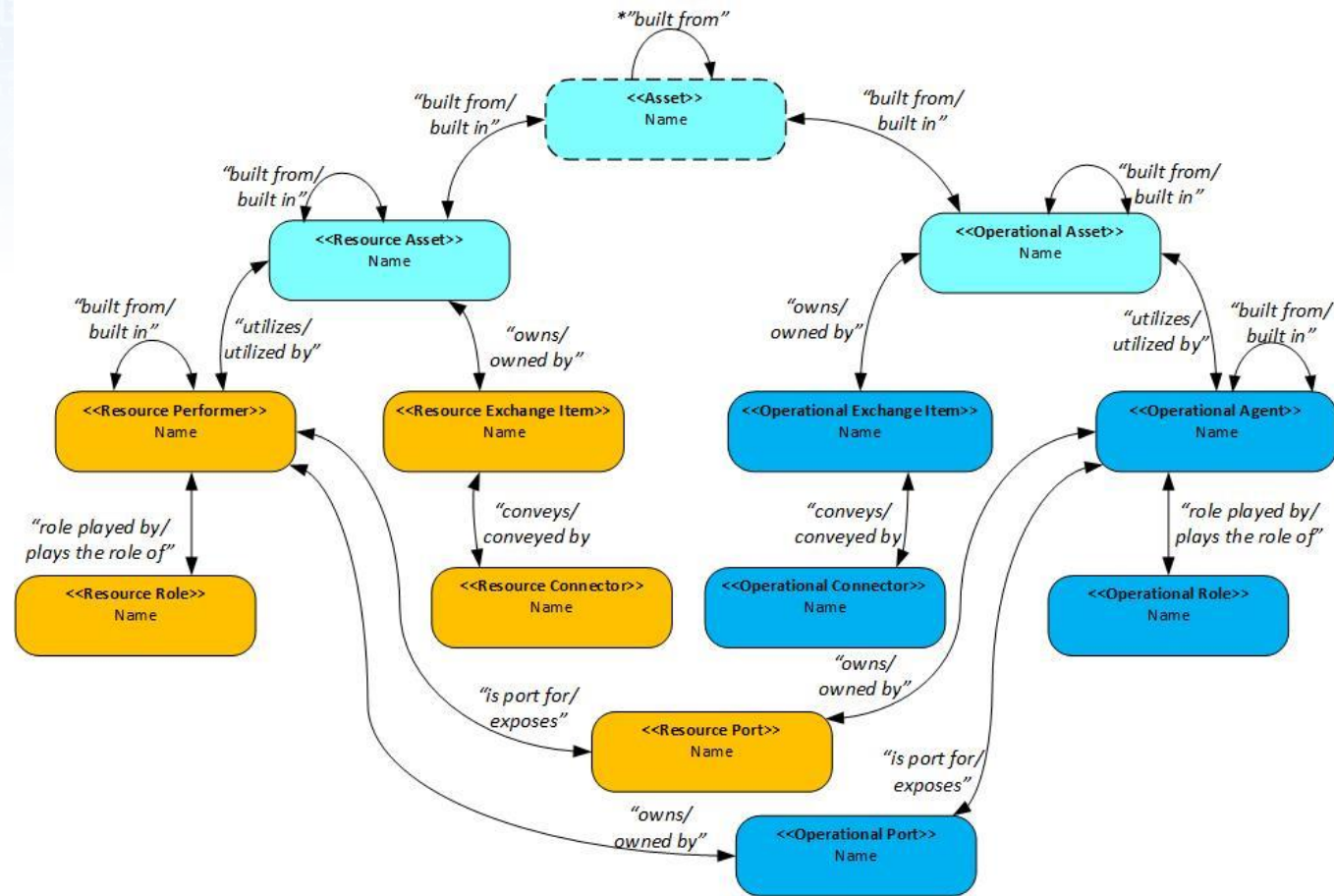
Security Motivation (Sc-Mv)



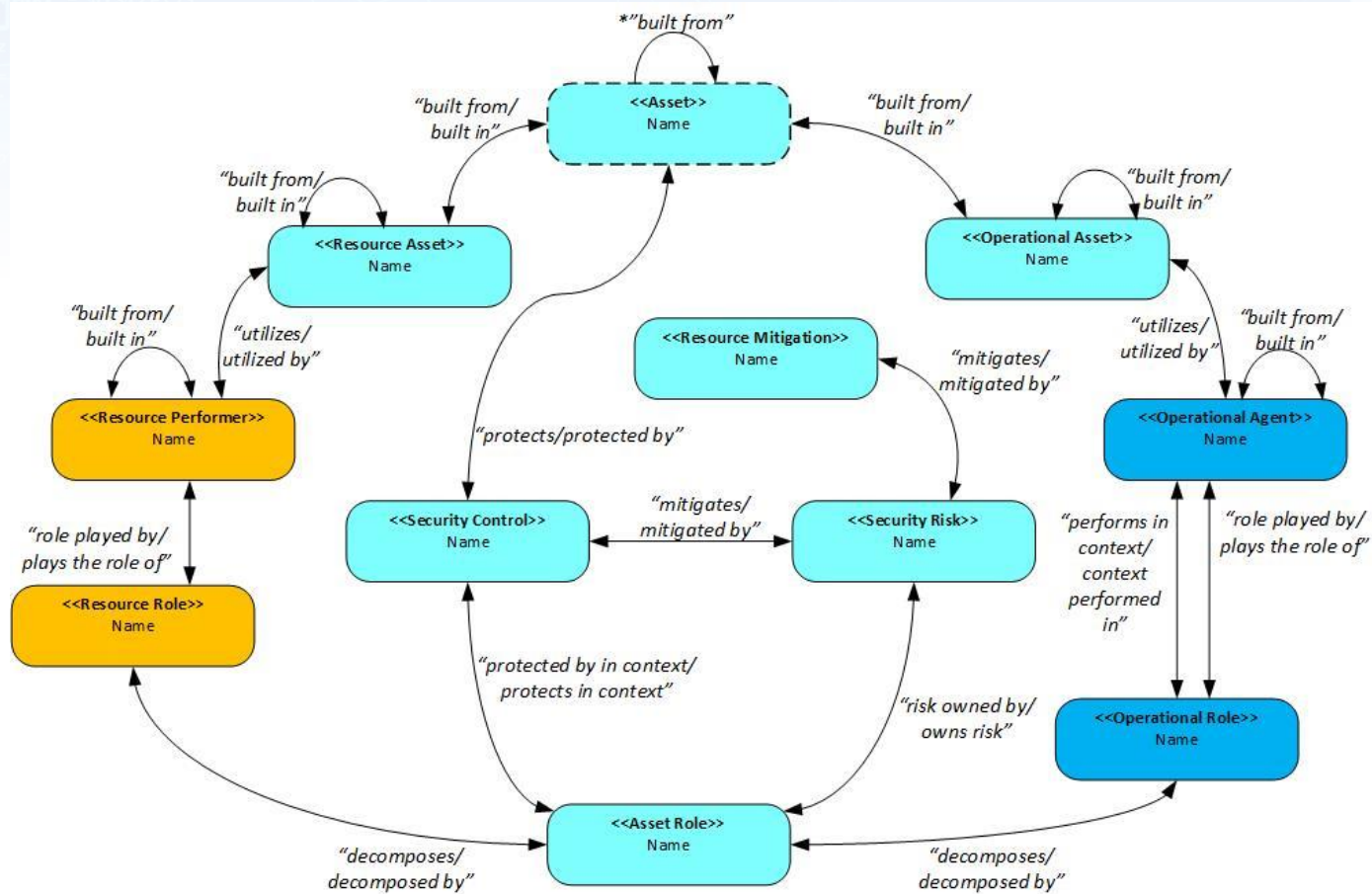
Security Taxonomy (Sc-Tx)



Security Structure (Sc-Sr)

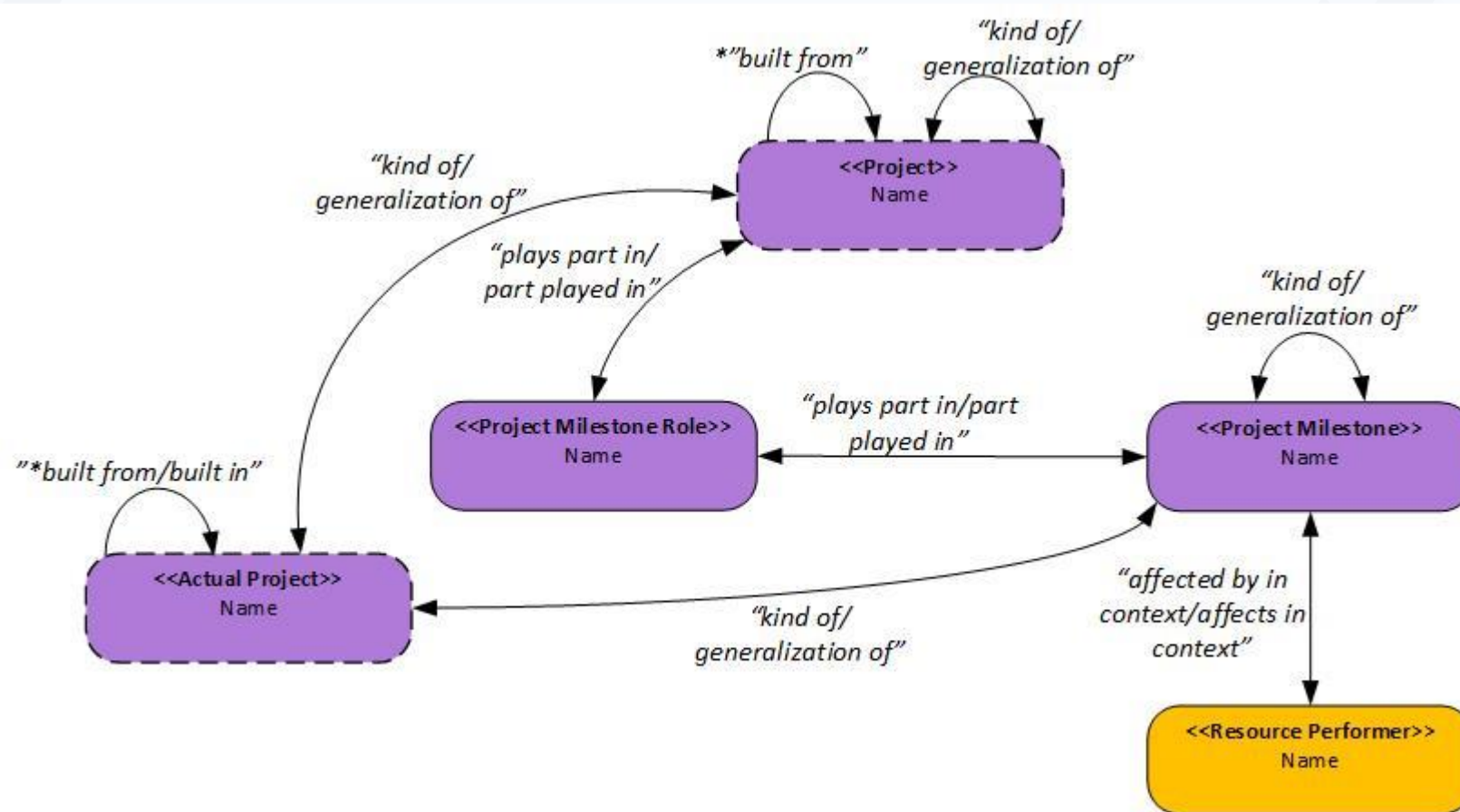


Security Traceability (Sc-Tr)

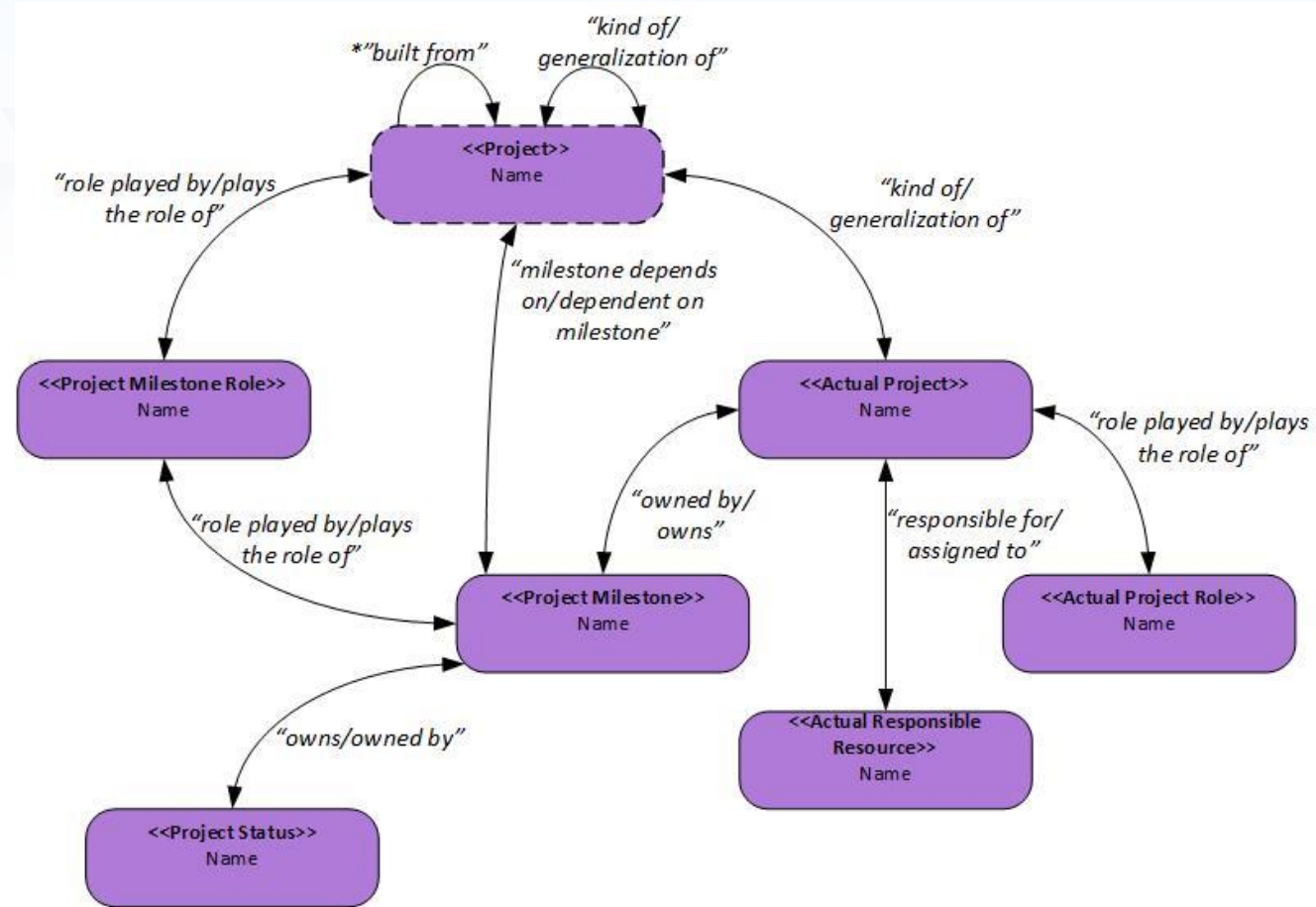


Projects (Pr)

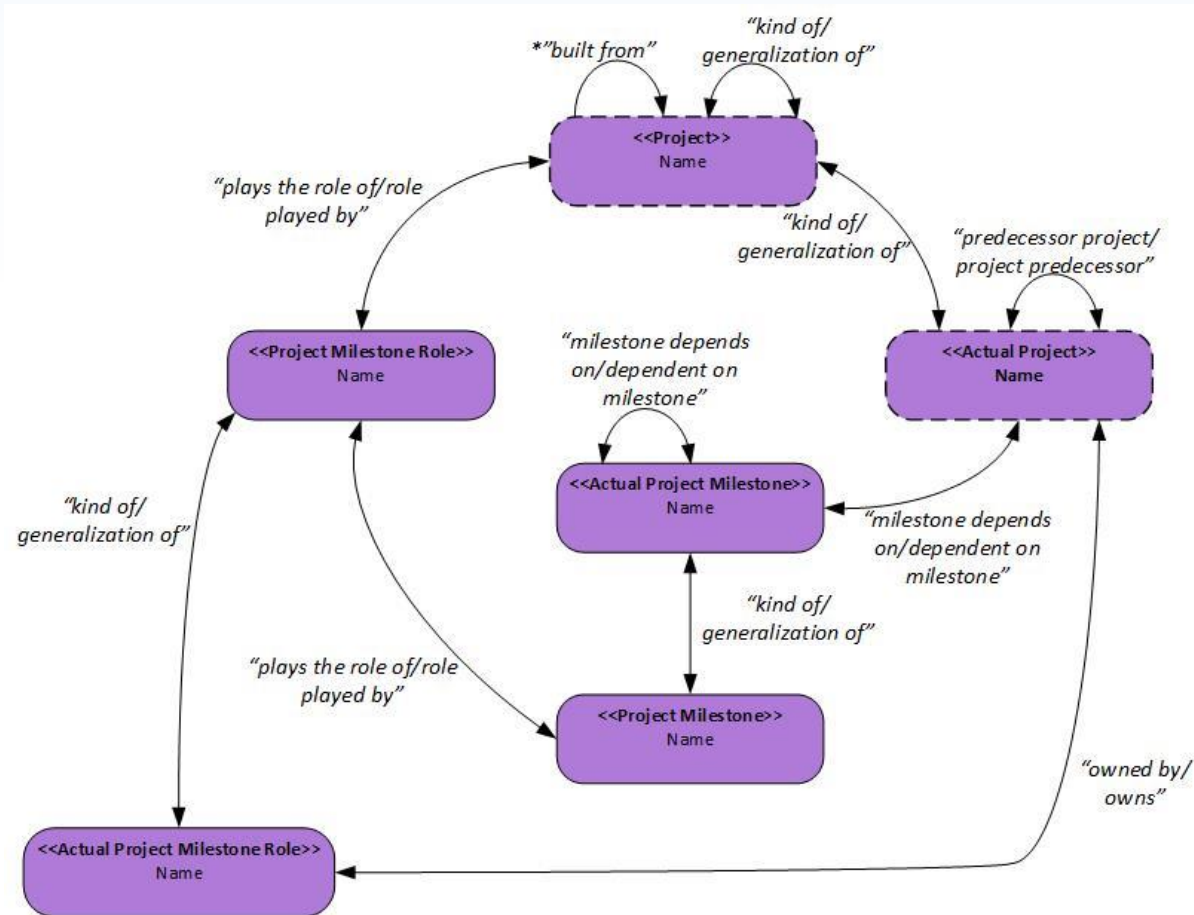
Projects Taxonomy (Pj-Tx)



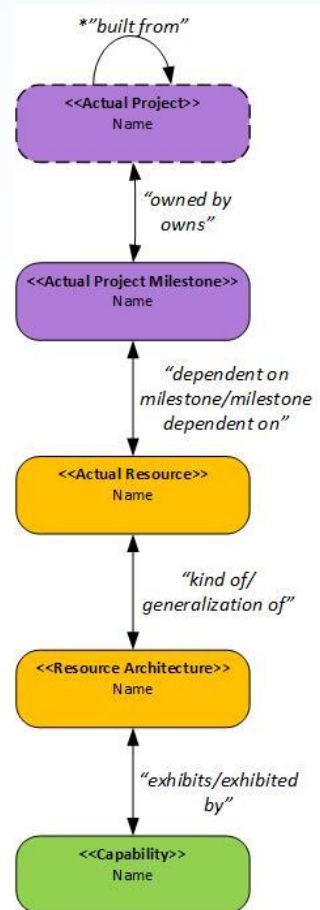
Projects Structure (Pj-Sr)



Projects Connectivity (Pj-Cn)

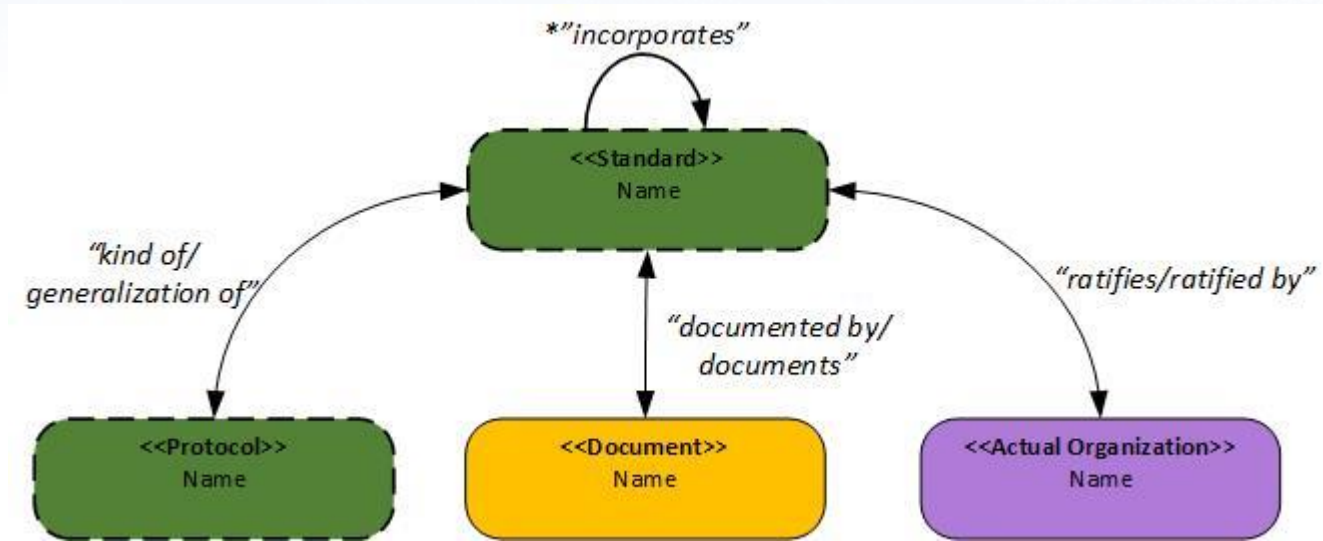


Projects Traceability (Pj-Tr)



Standards (SD)

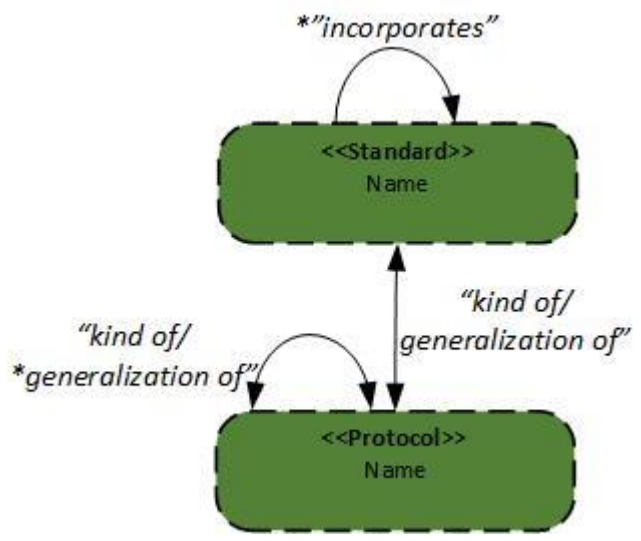
Standard Taxonomy (Sd-Tx)



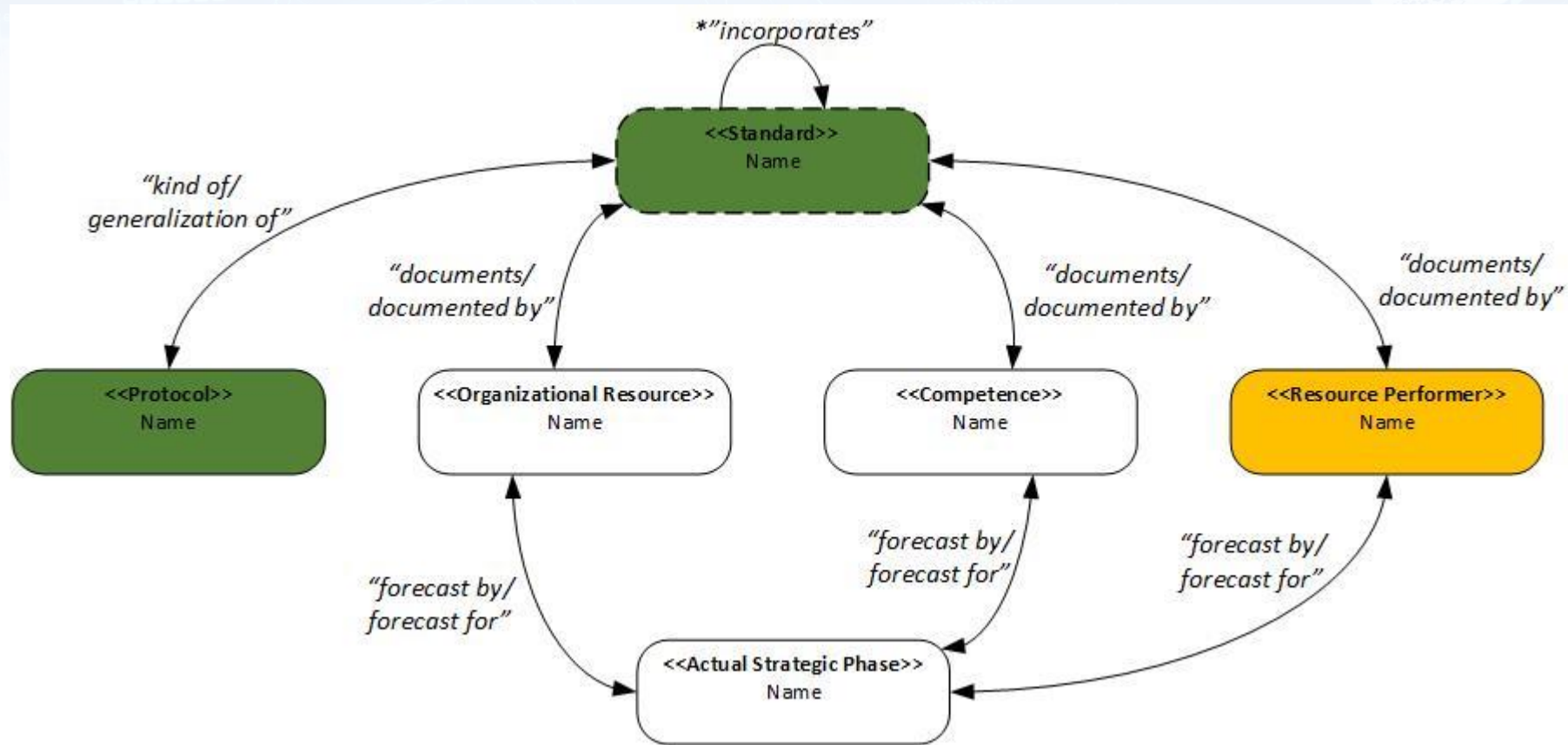
Standard Structure (Sd-Sr)

```
#define ASM_VRX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"  
static __always_inline unsigned long v  
    g_value_t (<std>)  
}  
{  
    lcs_solution lcs_half(const <std>, const string &b)  
    lcs_solution result ();  
    const size_t size = a.size() + b.size();  
    for (size_t i = 0; i < size; ++i)  
        (on (size_t) = 0; i < size; ++i)  
        if (a[i] < b[i])  
            result[i] = a[i];  
        else  
            result[i] = b[i];  
    return result;  
}
```

```
%include "win32n.inc"  
extern MessageBoxA  
import MessageBoxA user32.dll  
extern WINAPI  
extern WINAPI  
extern WINAPI
```



Standards Roadmap (Sd-Rm)



Standards Traceability (Sd-Tr)

```
#define ASM_VRX_VMREAD_RDX_RAX    ".byte 0xDf, 0x7d, 0xd0"
```

```
static __always_inline unsigned long vpx
```

```
    u_value; /*(smt)*/
```

```
    lcs_solution lcs_half(const <math>SMTProblem</math> &const string &b)
```

```
    lcs_solution result ();
```

```
    const size_t size = a.size() + b.size();
```

```
    for (size_t i = 0; i < size; ++i)
```

```
        for (size_t j = 0; j < size; ++j)
```

```
            lcs_solution lcs(a.substr(0, i), b.substr(0, j));
```

```
            if (i > 0 && j > 0 && a[i-1] == b[j-1])
```

```
                lcs_solution lcs2(a.substr(0, i-1), b.substr(0, j-1));
```

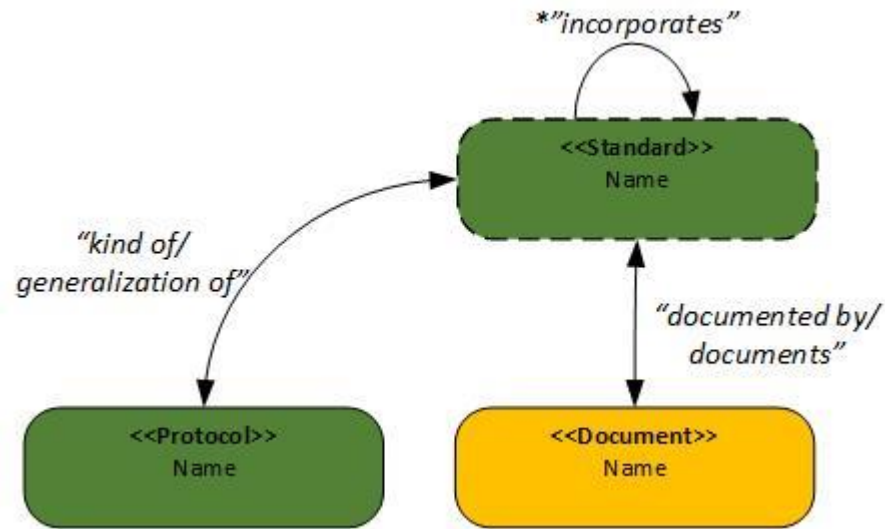
```
%include "win32n.inc"
```

```
extern MessageBoxA
```

```
__declspec(dllexport) void user32_dll
```

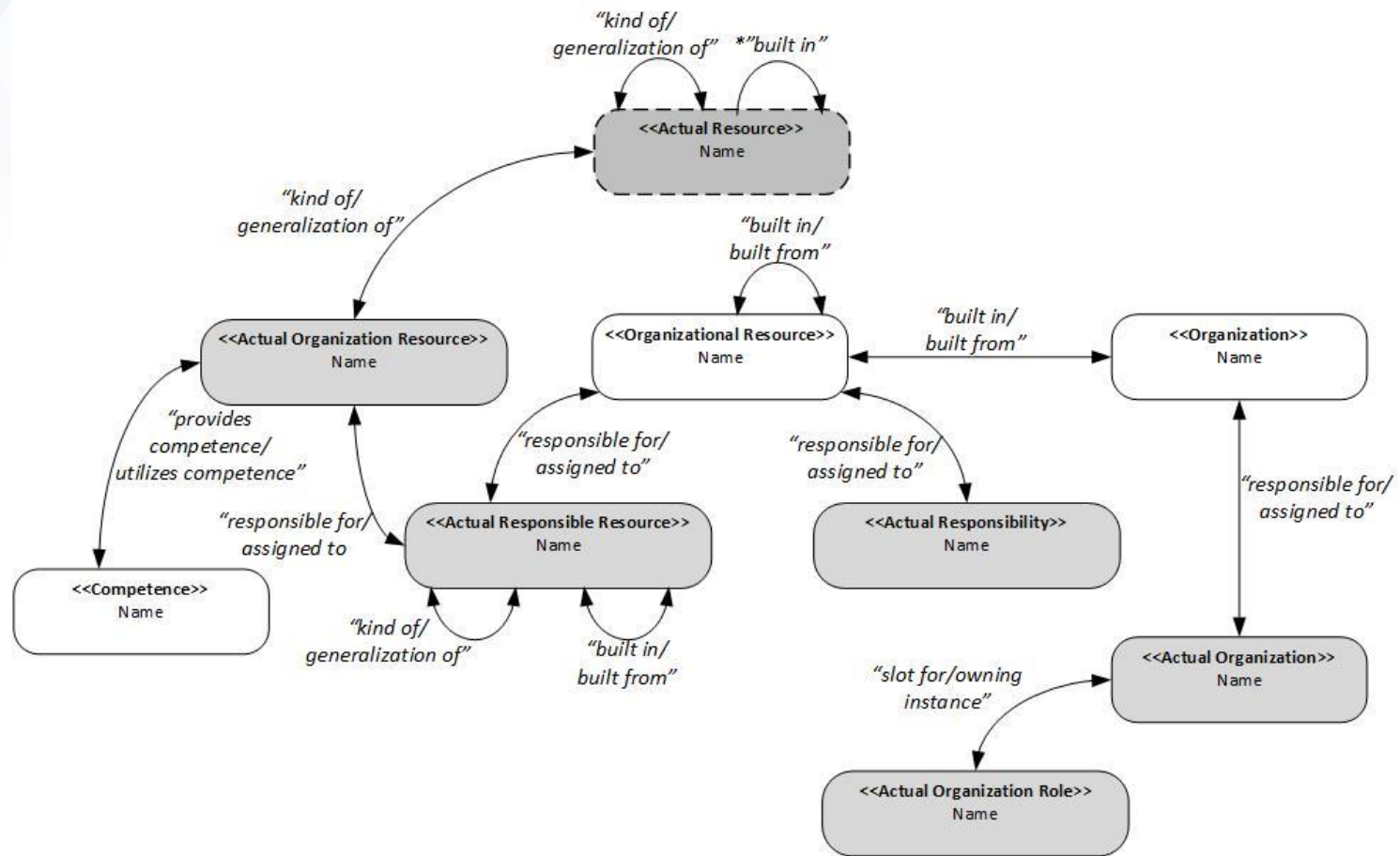
```
__declspec(dllexport) void
```

```
__declspec(dllexport) void user32_dll
```

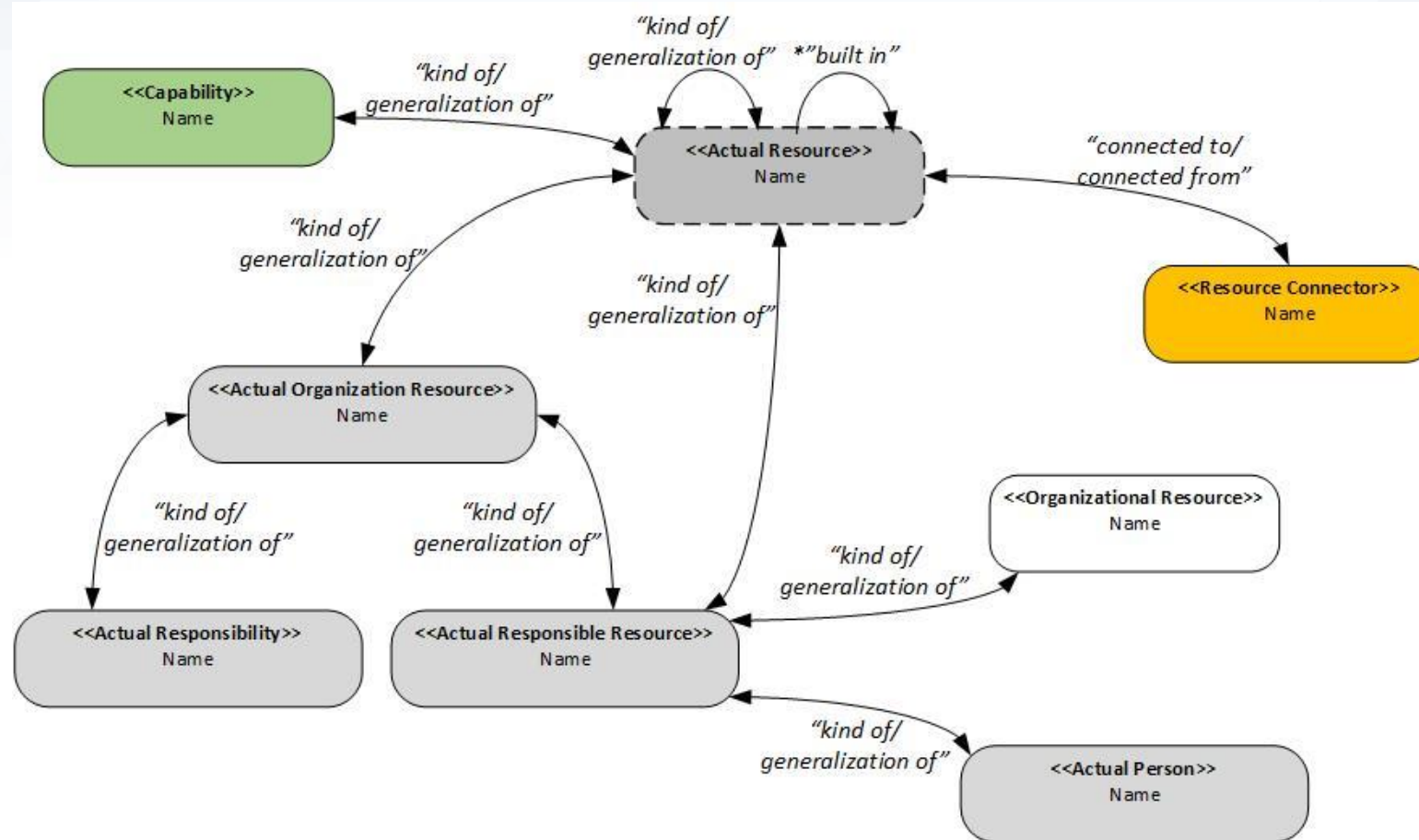


Actual Resources (Ar)

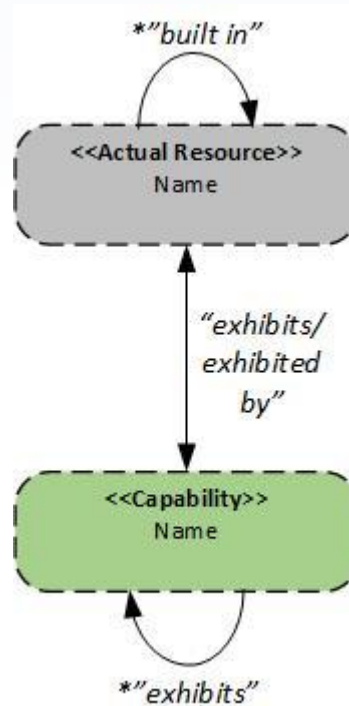
Actual Resources Structure (Ar-Sr)



Actual Resources Connectivity (Ar-Cn)

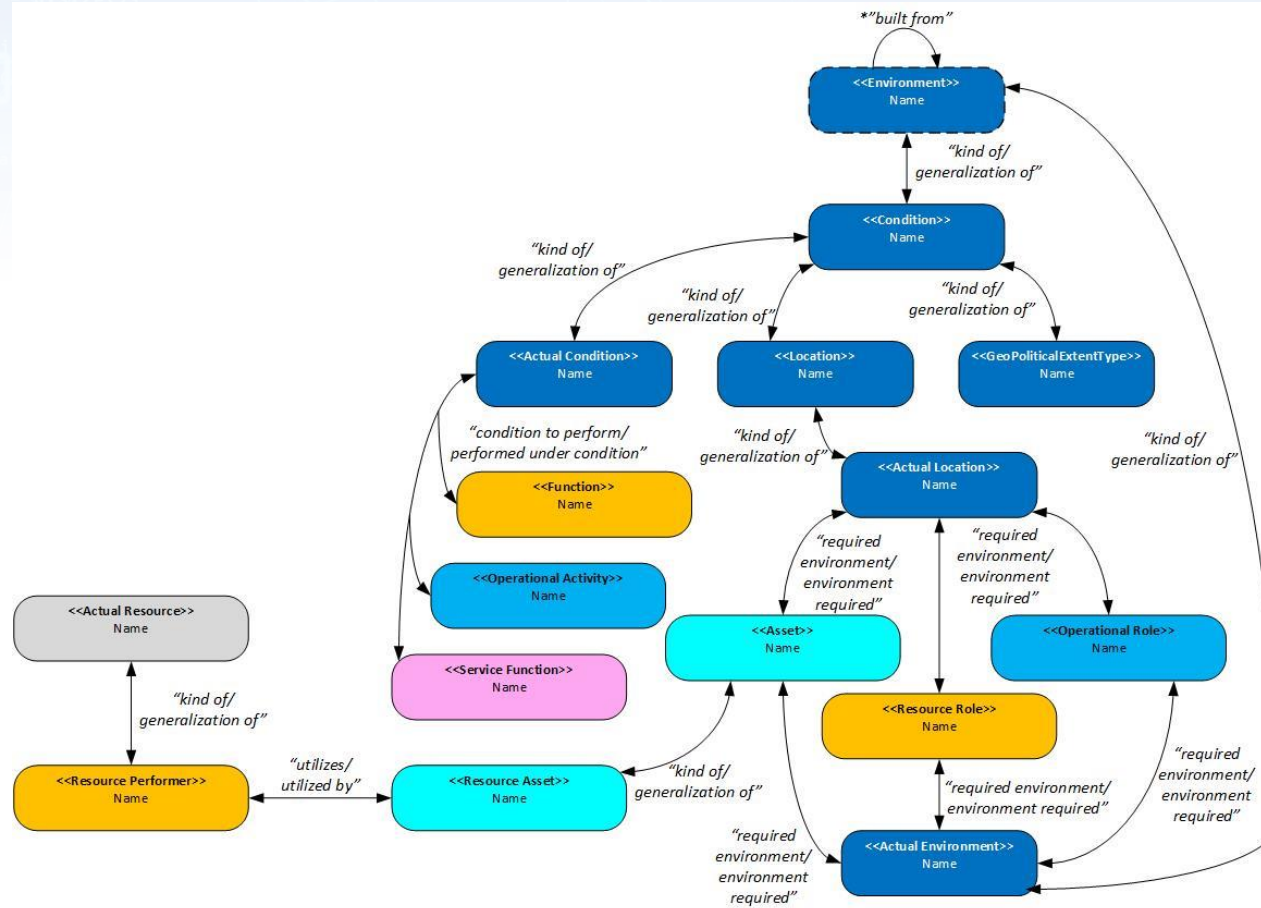


Actual Resources Traceability (Ar-Tr)

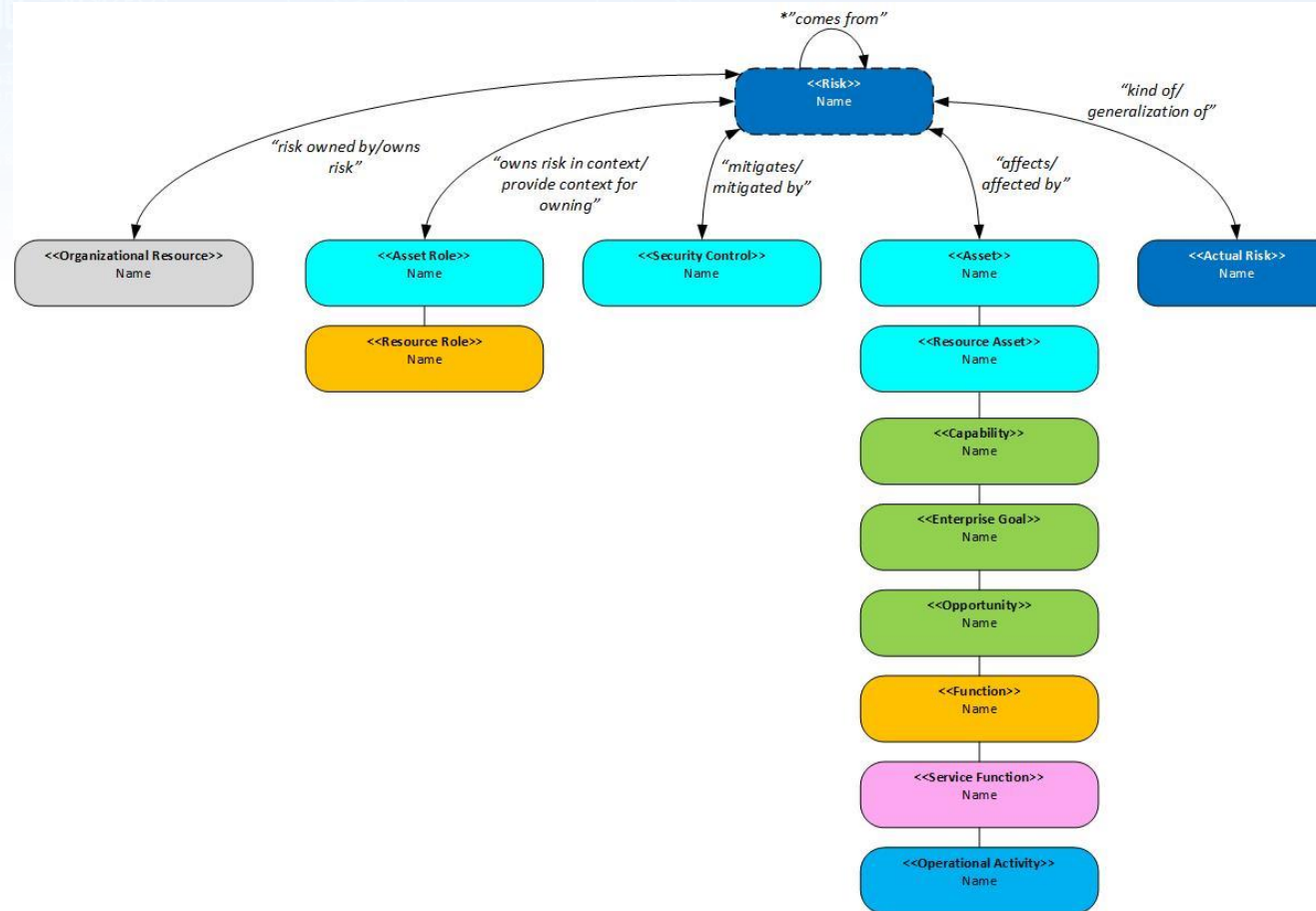


Parameters (Pm)

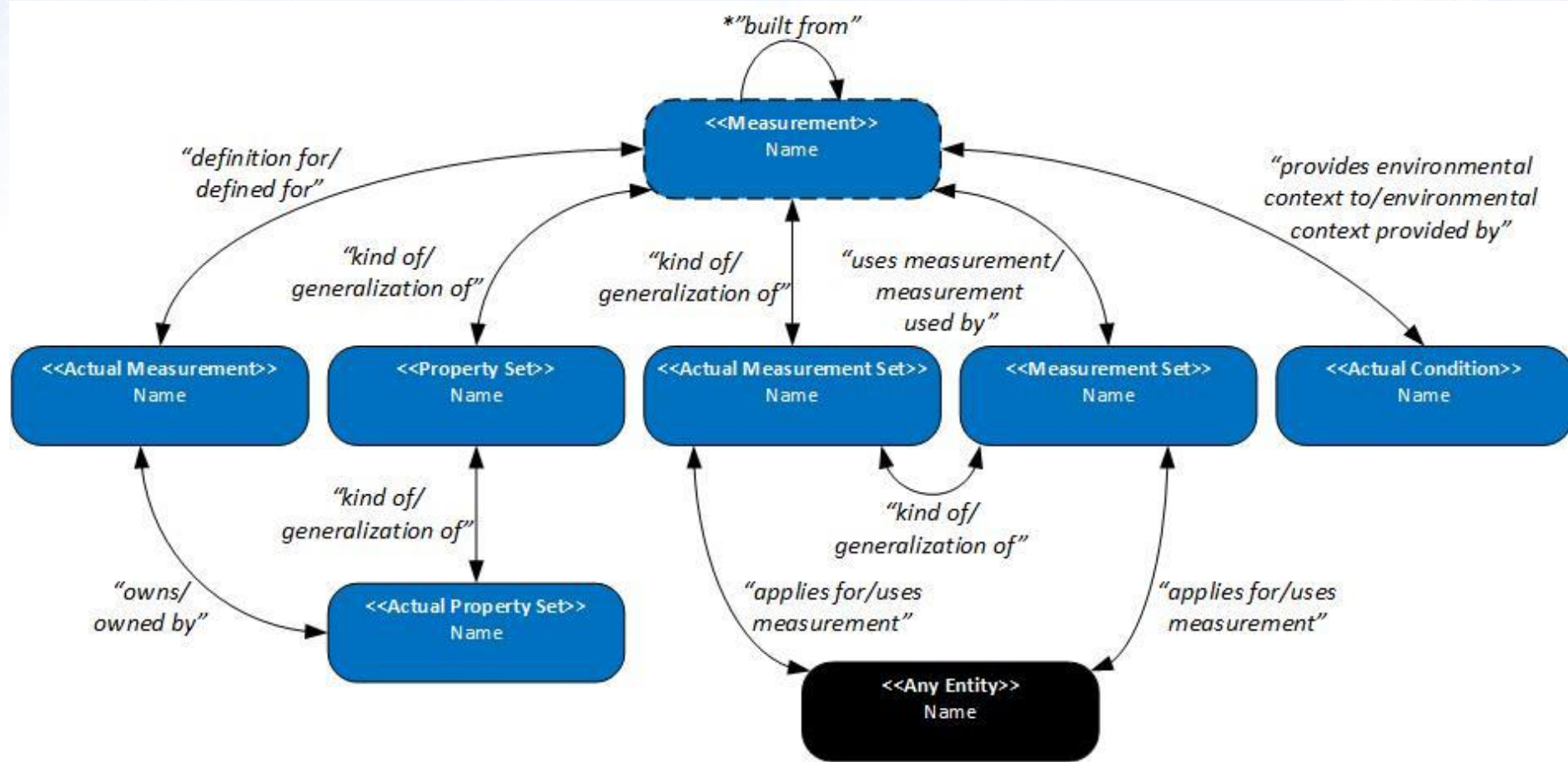
Environment (En-Pm-E)

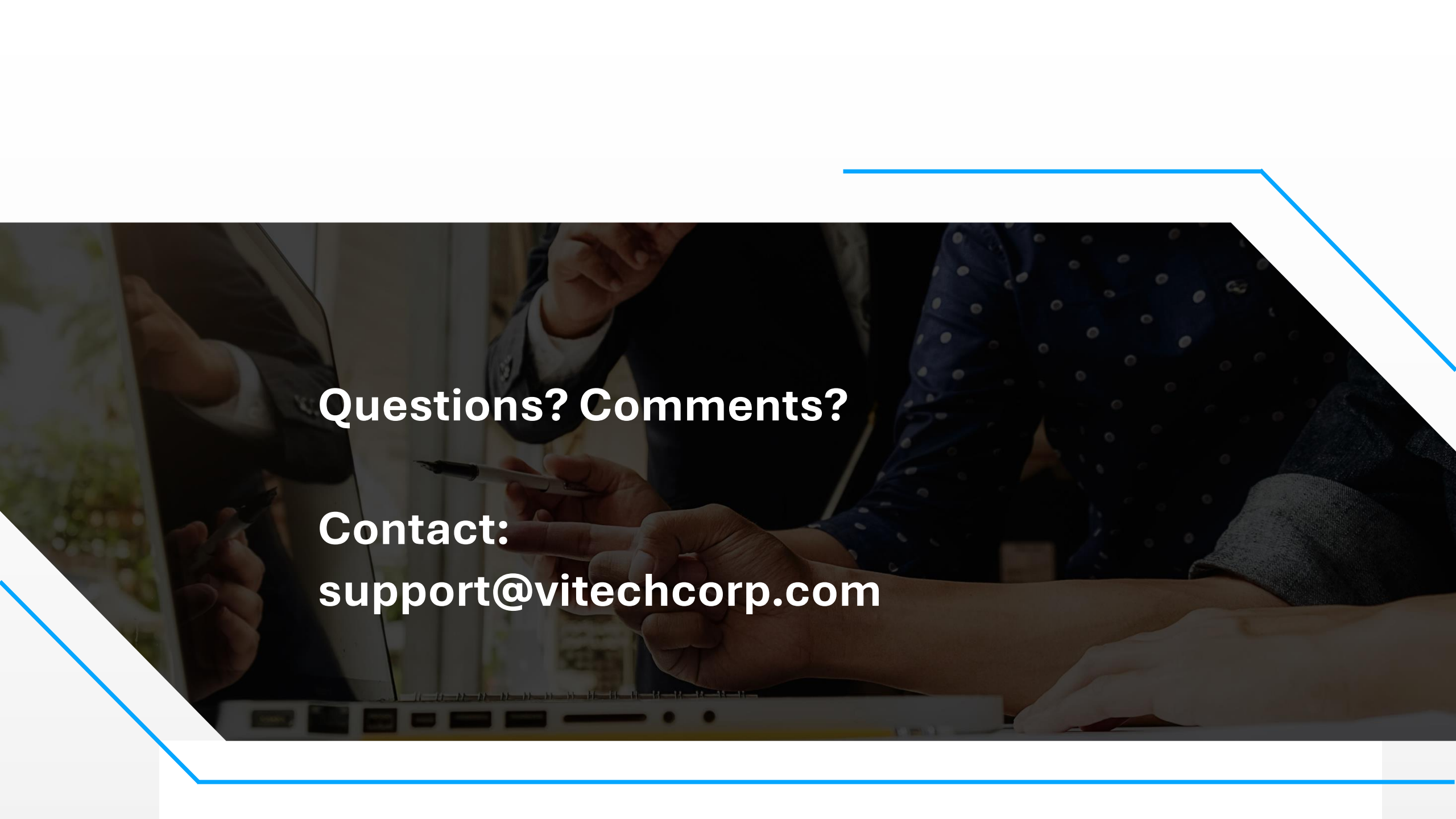


Risks (Rk-Pm-R)



Measurements (Me-Pm-M)





Questions? Comments?

Contact:
support@vitechcorp.com